A Study of the SHA-2 Cryptographic Hash Family

Somitra Kumar Sanadhya

Applied Statistics Unit
Indian Statistical Institute
Kolkata
February 2009

A Study of the SHA-2 Cryptographic Hash Family

Somitra Kumar Sanadhya

Thesis submitted to the Indian Statistical Institute
in partial fulfillment of the requirements
for the award of
Doctor of Philosophy

Thesis supervisor: Prof. Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata.

List of publications

Journal Paper

1. Somitra Kumar Sanadhya and Palash Sarkar. A Combinatorial Analysis of Recent Attacks on Step Reduced SHA-2 Family. *Cryptography and Communications - Discrete Structures, Boolean Functions and Sequences*, Accepted for publication, 2009.

Conference Papers

2. Somitra Kumar Sanadhya and Palash Sarkar. New Local Collisions for the SHA-2 Hash Family. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings*, volume 4817 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2007.

3. Somitra Kumar Sanadhya and Palash Sarkar. Attacking Reduced Round SHA-256. In Steven Bellovin and Rosario Gennaro, editors, *Applied Cryptography and Network Security - ACNS 2008, 6th International Conference, New York, NY, June 03-06, 2008, Proceedings*, volume 5037 of *Lecture Notes in Computer Science*, pages 130–143. Springer, 2008.

4. Somitra Kumar Sanadhya and Palash Sarkar. Non-Linear Reduced Round Attacks Against SHA-2 Hash family. In Yi Mu and Willy Susilo, editors, *Information Security and Privacy - ACISP 2008, The 13th Australasian Conference, Wollongong, Australia, 7-9 July 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2008.

5. Somitra Kumar Sanadhya and Palash Sarkar. Deterministic Constructions of 21-Step Collisions for the SHA-2 Hash Family. In Editors, Tzong-Chen Wu and Chin-Laung Lei and Vincent Rijmen and Der-Tsai Lee, *Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 2008, Proceedings*, volume 5222 of

Lecture Notes in Computer Science, pages 244–259. Springer, 2008.[1]

6. Somitra Kumar Sanadhya and Palash Sarkar. New Collision Attacks Against Up To 24-step SHA-2 . In D.R. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, December 2008, Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 91–103. Springer, 2008.

7. Somitra Kumar Sanadhya and Palash Sarkar. A New Hash Family Obtained by Modifying the SHA-2 Family . In Rei Safavi-Naini and Vijay Varadharajan, editors, *ASIACCS 2009, 4th International Conference, March 10-12, 2009, Sydney, Australia*, volume To appear of *ACM Conference Proceedings*, ACM, 2009.

[1]This paper received one of the three best student paper awards at the conference.

Dedicated to my teachers.

Prof. V.B. Deshpande, IIT Delhi,
Prof. S. Balasundaram, JNU New Delhi,
Prof. Palash Sarkar, ISI Kolkata.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Palash Sarkar for introducing me to the fascinating and challenging world of hash function cryptanalysis. Whatever little I know about the rich subject of cryptology is primarily through him.

The crypto group at the Indian Statistical Institute (ISI) has been a part of my extended family in the last few years. I do not have words to thank its leaders Prof. Bimal Roy, Prof. Subhamoy Maitra and Prof. Rana Barua, apart from Prof. Palash Sarkar, for all the academic and non-academic help that they have extended to me during my stay in Kolkata.

I am thankful to all the student members of our crypto lab for maintaining lively research atmosphere there. I had a great time working in the lab, where I spent more time than at my home in last few years. I am also thankful to the Ministry of Information Technology, Govt. of India for supporting my academic visits. I take this opportunity to thank Applied Statistics Unit (ASU) for providing me computing facilities and providing me a place to work. All the staff of ASU was very cooperative and helpful and I thank them all.

My little sons Ved and Raghav have been a great source of inspiration. Life would not have had the same meaning without their smile. Playing with them has been an efficient and joyous substitute for my physical exercises.

I am indebted to my loving father and late mother for giving me the courage to pursue my interests in life and for making me an independent person. I am also indebted to my father-in-law and mother-in-law for all the support while I remained engaged with research work. Their moral support always remained a guiding force to me.

I was fortunate to have had some great teachers from whom I learned much more than the subjects they expertised in. A *guru* is placed above God In Indian tradition and they showed me why. I dedicate this work to three of my most loving teachers – Prof V. B. Deshpande, Prof. S. Balasundaram and Prof. Palash Sarkar.

Finally, I am thankful to my wife Geetanjali, who sacrificed her personal life so that I could pursue research. My research as well as personal life owes a lot to her. Her love and emotional support enabled me to remain strong in all circumstances. I do not have words to thank her. I only wish that our love and faith in each other grows stronger with each passing year together.

February 2009,
Kolkata.

Contents

Notation

- The word size n is 32 for SHA-256 and 64 for SHA-512.
- The number of steps r is 64 for SHA-256 and 80 for SHA-512. We use 'step' and 'round' synonymously.
- Message words: $W_i \in \{0,1\}^n$, $W'_i \in \{0,1\}^n$ for any i.
- X^j: j^{th}-bit of the word X.
- Message pair: $\{W_0, W_1, W_2, \ldots W_{15}\}$ and $\{W'_0, W'_1, W'_2, \ldots W'_{15}\}$.
- Expanded message pair: $\{W_0, W_1, W_2, \ldots W_{r-1}\}$ and $\{W'_0, W'_1, W'_2, \ldots W'_{r-1}\}$.
- The internal registers for the message pair in step i: $\text{REG}_i = \{a_i, \ldots, h_i\}$ and $\text{REG}'_i = \{a'_i, \ldots, h'_i\}$.
- $\text{ROTR}^k(x)$: Right rotation of an n-bit quantity x by k bits.
- $\text{SHR}^k(x)$: Right shift of an n-bit quantity x by k bits.
- $\overline{x}$: Bitwise negation of x, where x can be an n-bit or a 1-bit quantity.
- $*$: Logical 'AND' (i.e. multiplication) of 2 single bits.

Note: In this thesis we use two kinds of differences – XOR based and modular subtraction based. We use Δ for XOR differences and δ for modular differences.

- $\oplus$: bitwise XOR.
- $+$: addition modulo 2^n, when operands are n-bit quantities.
- $-$: subtraction modulo 2^n, when operands are n-bit quantities.
- $\Delta X = X' \oplus X$ where X is an n-bit quantity.
- $\delta X = X' - X$ where X is an n-bit quantity.

- Σ_0, Σ_1: n-bit to n-bit functions used in the compression function of SHA-2 family members. They are defined differently for SHA-256 and SHA-512. For details refer to Section 1.4.
- σ_0, σ_1: n-bit to n-bit functions used in the message expansion of SHA-2 family members. They are defined differently for SHA-256 and SHA-512. For details refer to Section 1.4.
- $\delta\sigma_0(\delta W_i)$: Output difference of the σ_0 function when its inputs differ by δW_i. That is, $\delta\sigma_0(\delta W_i) = \sigma_0(W_i') - \sigma_0(W_i) = \sigma_0(W_i + \delta W_i) - \sigma_0(W_i)$.
- $\delta\sigma_1(\delta W_i)$: Output difference of the σ_1 function when its inputs differ by δW_i. That is, $\delta\sigma_1(\delta W_i) = \sigma_1(W_i') - \sigma_1(W_i) = \sigma_1(W_i + \delta W_i) - \sigma_1(W_i)$.
- $\delta\Sigma_1^i(x)$: Output difference of the Σ_1 function in step i when its inputs differ by x. That is, $\delta\Sigma_1^i(x) = \Sigma_1(e_i') - \Sigma_1(e_i) = \Sigma_1(e_i + x) - \Sigma_1(e_i)$.
- $\delta\Sigma_0^i(x)$: Output difference of the Σ_0 function in step i when its inputs differ by x. That is, $\delta\Sigma_0^i(x) = \Sigma_0(a_i') - \Sigma_0(a_i) = \Sigma_0(a_i + x) - \Sigma_0(a_i)$.
- $\delta f_{MAJ}^i(x, y, z)$: Output difference of the f_{MAJ} function in step i when its inputs differ by x, y and z. That is, $\delta f_{MAJ}^i(x, y, z) = f_{MAJ}(a_i', b_i', c_i') - f_{MAJ}(a_i, b_i, c_i) = f_{MAJ}(a_i + x, b_i + y, c_i + z) - f_{MAJ}(a_i, b_i, c_i)$.
- $\delta f_{IF}^i(x, y, z)$: Output difference of the f_{IF} function in step i when its inputs differ by x, y and z. That is, $\delta f_{IF}^i(x, y, z) = f_{IF}(e_i', f_i', g_i') - f_{IF}(e_i, f_i, g_i) = f_{IF}(e_i + x, f_i + y, g_i + z) - f_{IF}(e_i, f_i, g_i)$.

Abbreviations

- NIST: National Institute of Standards and Technology.
- SHA: Secure Hash Algorithm.
- LC: Local Collision. For details, refer to Section 1.6.
- DP: Differential Path. For details, refer to Section 1.7.
- DV: Disturbance Vector. For details, refer to Section 1.8.
- MSB: Most Significant Bit.
- LSB: Least Significant Bit.
- GH Local Collision: The Gilbert and Handschuh local collision [22].
- NB Local Collision: The Nikolić and Biryukov local collision [38].

LC(X): Denotes the 9-step message differentials as per the local collision when the initial message difference is an n-bit quantity X.

List of Figures

List of Tables

Introduction

Cryptology is the study of secret communication and information hiding. It covers two closely related branches – cryptography and cryptanalysis. Cryptography is the science of the enciphering and deciphering of messages in secret code or cipher. On the other hand, the science and art of recovering information from ciphers (without knowing some hidden details) is known as cryptanalysis.

Cryptology has been used in some form or the other since ages. Before the nineteenth century, cryptology, or more specifically cryptanalysis, was closely related to linguistics. There are many writings of the ancient people which have been lost. The best known example of such a language is available in the Egyptian "Hieroglyphics". The hieroglyphs contain the written language of thousands of years of Egyptian culture. These remained a mystery for nearly two thousand years until a tablet known as the Rosetta Stone was discovered by Napoleon's soldiers while making excavations necessary to build a fort. It was quickly recognized by Champollion, the leading Egyptologist at the time, that the Rosetta stone had the same passage written in hieroglyphics, Demotic, and ancient Greek. The last two being known languages, Champollion could prove his idea that the hieroglyphics were a phonetic alphabet for a language closely related to modern day Coptic. The Rosetta Stone was the key to the decipherment of hieroglyphics.

Another example of linguistic cryptanalysis is for the language "Linear B", which was the style of written language used in ancient Crete during (or possibly shortly thereafter) the supposed time of King Minos. Around 1940, at the age of eighteen, Michael Ventris discovered a precise correspondence between the glyphs of Linear B and the syllables of ancient Greek, and thereby enabled the decipherment of that language. This also proved that the Greeks had arrived in Crete at that time.

Despite the modern advances, some ancient languages remain hidden till date. Two of the prominent examples are Linear A (a language older than Linear B) and the language of the Indus valley.

The first recorded use of cryptography for correspondence is by the Spartans in $\approx$ *400 BC*. The Spartans employed a cipher device called a "scytale" to send secret communications between military commanders. The scytale consisted of a staff of wood around which a piece of papyrus or leather or parch-

ment was close-packed and inscribed with the message. Once unwrapped the parchment appeared to contain an incomprehensible set of letters, however when wrapped around another staff of identical size the original text appears [29].

Some Hebrew writings ($\approx$ *400 BC*) contain what is now called as the Atbash substitution cipher to jumble and hide the written message [29]. Few centuries later, around *100 BC*, Julius Caesar's army used alphabets shifted by 3 places cyclically to transfer messages.

An interesting use of secret communication is described in the *Kamasutra* of *Vatsyayana* ($\approx$ *400 AD*) where the intended use is to exchange messages between lovers while keeping it secret from others [13]. The text mentions "the ability to communicate secretly" as one of the sixty four arts that men and women should possess. However, no details about how this was to be achieved are available in the *Kamasutra*.

Greek historian Herodotus provides earliest records of hiding a message in some other medium, without actually ciphering the message. This technique is known as steganography. When Histiaeus had to send a secret message to his son-in-law, he shaved the head of a slave and tattooed a message, he waited till the hair had grown before dispatching him in order to avoid detection. The unfortunate slave was not allowed to take bath for few months. Another Greek history was when Demeratus scraped the wax off tablets and wrote messages on the underlying wood. He then covered the wood with wax again to conceal the message. The tablets appear to be blank and unused when inspected.

Invisible ink has always been a popular method of steganography. Ancient Romans wrote between lines using invisible inks made from substances like milk, urine and fruit juices. When it is heated, the invisible ink would darken and become legible.

Even in modern times, several special ways of writing, such as different spaces between characters or accents on some characters, have been used to denote certain characters in the written communication. Pictures with hidden messages, texts containing anagrams and micro-dots were also used in numerous situations including in the second world war. For an interesting and comprehensive discussion of the history of cryptography, see [29].

Modern Cryptology. The scientific basis for cryptology started with the legendary paper of Claude Shannon [55]. Prior to this work, cryptosystems were designed in an ad-hoc manner. Shannon laid the foundations for the theoretical analysis and design of cryptosystems.

The principal aims of modern cryptosystems are confidentiality, integrity and authorization. Confidentiality is achieved by the use of block or stream ciphers, integrity is usually achieved by the use of hash functions and authorization is usually achieved by access control lists. Block ciphers, stream

ciphers and hash functions are also used as primitives in numerous protocols to achieve special cryptographic needs, such as secure evaluation of a multi-party function etc. For further details about the primitives of cryptology, refer to [36, 61].

After Shannon's fundamental work, another milestone was achieved in the history of cryptology in 1976 by the work of Diffie and Hellman [16]. Prior cryptosystems required the use of a secret key between every pair of parties which were interested in communicating secretly. This resulted in problems of key distribution, since n interested parties would require C_2^n secret keys. The work of Diffie and Hellman showed that it was possible to design cryptosystems in which both the communicating parties *do not* share the *same* secret key. In their setting, each party is supposed to have a "public key" and a corresponding "private key". The public key of a party is made available to everyone wishing to communicate with that party, while the private key is kept secret by the party for own use. This new design results in only n pair of keys for n interested parties.

The interesting thing about the new cryptosystem is that despite the keys used for ciphering and deciphering the message being different, they still result in perfectly inverse operation. Due to the different keys used at the two ends of communication, this type of cryptosystems were called "Asymmetric key cryptosystems" or "Public key cryptosystems". The earlier cryptosystems were, in retrospect, given the name "Symmetric key cryptosystems" or "Private key cryptosystems".

Although Diffie and Hellman showed that it was possible to design an asymmetric key cryptosystems, they did not instantiate such a cryptosystem. The first asymmetric cryptosystem was designed by Rivest, Shamir and Adleman in 1977. This is known as the RSA Cryptosystem [45]. For further details and technical information about the asymmetric key cryptosystems, refer to [36, 61].

Hash Functions. Cryptographic hash functions are an important primitive used in a variety of ways in modern cryptography. Essentially, a hash function produces a "digest" or a "fingerprint" of fixed length for a message of any arbitrary length [36]. Although all practical hash functions impose an upper bound on the length of the message, this bound is so large ($\geq 2^{64}$ bits) that for all practical purposes, one can think of a hash function as producing digest of any arbitrary sized message.

The hash function compression is necessarily "lossy" and it is not possible to reconstruct the original message from the digest alone. In this respect, the hash value of a message is somewhat like human fingerprints. One can compare a person's fingerprint with a recorded fingerprint and ascertain the person's identity; but by seeing only a fingerprint, not much can be learned about the identity of the person to whom it belongs. Mathematically, a hash

function is a mapping from a very large domain to a much smaller range.

A hash function is required to satisfy some properties in order to be useful for cryptographic applications. Some of the important ones among these are "collision resistance", "pre-image resistance" and "second pre-image resistance". In addition, some "randomness" properties are also required to be satisfied from a hash function. Loosely speaking, this means that even if one knows a message and its hash value, it should be difficult to predict the bits of the hash value for another message. In some sense, the bits of the hash value should appear random.

A collision resistant hash function has the property that it is *infeasible* to develop an efficient algorithm to produce two distinct messages which will produce the same fingerprint (hash value computed by the hash function under consideration) for these two different messages. Note that it is *impossible* to construct a hash function which will completely eliminate these "hash collisions". This is due to the fact that a hash function has much larger domain than the range.

The presence of collisions notwithstanding, the design of a hash function attempts to make finding such collision computationally difficult. One of the design goals of hash functions is that finding collision for it should not be easier than the "birthday attack" [36].

A pre-image resistant hash function makes it computationally difficult to find (one among the many possible) pre-images for a given hash value. The second pre-image resistance property is similar to the collision resistance in that the aim is to find two different messages both hashing to the same value. However, the difference is that for second pre-image resistance, one of the messages is given and the other colliding message is to be found.

Generic Attacks Against Hash Functions. There are two generic attacks which are applicable to any hash function. These are described next.

1. *Brute force attack.* Since a hash function maps a large (but finite) domain to a much smaller sized co-domain, any random element in the co-domain will likely have multiple pre-images. Repeatedly trying to hash random messages will ultimately yield a pre-image (or a second pre-image) for any random (or chosen) hash value. This attack is the so called "brute force" attack. Let H be a hash function defined as $H : \mathcal{X} \rightarrow \mathcal{Y}$. Then the complexity of the brute force attack is $2^{|\mathcal{Y}|}$ for both the pre-image attack and for the second pre-image attack for H.

2. *Birthday attack.* The birthday attack is a generic attack against the collision resistance of a hash function. It stems from the so called "Birthday paradox". The birthday paradox is not really a paradox, but merely a surprising statistical fact. It states that if there are more

than 23 people in a group, then the probability of two people in the group sharing the same birthday is more than 1/2.

This simple statistical fact has important consequences for the design of hash functions. Let H be a hash function defined as above. Then a collision is expected for H in roughly $2^{|\mathcal{Y}|/2}$ evaluations of H. What it means for H is that even though H may produce $|\mathcal{Y}|$-bit digests, it will take only $2^{|\mathcal{Y}|/2}$ effort to find two message with the same hash value. This fact provides a generic algorithm, known as the birthday attack, for finding a colliding pair of messages. The idea is to randomly generate messages to the order of square root of the co-domain size. Among them, with high probability, there is going to be a pair which will hash to the same digest. This implies that when digest of a hash function is of size $|\mathcal{Y}|$-bits, the hash function provides only $|\mathcal{Y}|/2$-bit security against collision attacks.

The practical significance of the birthday attack is that all hash functions are designed with sufficiently large co-domains. First widely used practical hash function MD4 was designed with co-domain size of 2^{128} bits. More recent hash functions such as SHA-1 and SHA-2 family have even larger co-domains.

"Breaking" of a Hash Function If an algorithm can be constructed which finds pre-images/ second pre-images/ collisions for a hash function in effort less than the generic attacks described above, then that hash function is said to be broken. This "academic break" may not immediately render the hash function obsolete, but it does provide a strong justification for replacing it from practical applications. We discuss this issue in some more details a little later.

In this thesis, we are concerned only with the collision attacks, hence we will refer to algorithms in comparison to the birthday attack. That is, if there exists an algorithm which generates a pair of different messages which collide for the hash function and the cost of such an algorithm is less than the birthday attack, then the hash function is said to be broken. For example, for a hash function which has 128-bit output, the existence of a collision producing algorithm with effort less than 2^{64} will render that hash function "broken".

Practical Use of a Hash Function: Digital Signature Scheme. Digital signatures are a fundamental primitive used in cryptology. They are used to achieve authentication, authorization and non-repudiation in various cryptographic protocols. The basic purpose of a digital signature is to provide a way to bind an entity with some information [36].

Digital signatures in the digital world serve the same purpose as physical signatures on paper in our day to day life. An authority provides a paper document putting its "seal" or "signature" on the document. This seal serves to authenticate the veracity of the document. Digital signatures are used to electronically "sign" documents (such as email, electronic bank transfer instructions etc.) for the same purpose.

However, there are some differences between the paper signatures and the digital signatures. Paper signatures are difficult to copy whereas any electronic information can be easily copied. Moreover, electronic information can be *exactly* duplicated. We explain below how this affects digital signatures.

Suppose a person buys an article online and pays for it by means of an electronic cheque. This electronic cheque can be encashed by the merchant with the bank. The bank pays the merchant once it verifies that the signature indeed belongs to the customer.

There are few problems with this setting. Since any electronic information is easy to copy, the merchant can change the amount payable to him on the electronic cheque. Worse still, the merchant can repeatedly use the same instruction to collect money from the bank. Scenarios such as these cause a basic difference in the way digital signatures are used. Unlike paper signatures which are the same on all documents, digital signatures change as the document changes. Thus, a digital signature is not unique to a person, but rather unique to a {person, document} pair.

Creating and Verifying Digital Signatures Digital signatures are created by using a public key cryptosystem. We briefly review the essential idea of a public key cryptosystem. For further details, refer to [36, 61].

A public key cryptosystem utilizes a pair of keys – a public key `pk` and a secret key `sk`, corresponding to each party which wishes to receive confidential information. The encryption function $E_{pk}(m)$ encrypts message m corresponding to the public key into cipher text c (say). The decryption function $D_{sk}(c)$ decrypts the cipher text back to the plain text m. The cryptosystem relies on the fact that it is not feasible to compute sk even when pk is known.

We explain the basic idea of RSA based digital signature scheme next. A message m is first hashed to a quantity $h(m)$ using a hash function h. This hash value $h(m)$ is encrypted by the private key, say sk, of the sender. Finally, both the message m and the digital signature $E_{sk}(h(m))$ are sent to the receiver. Upon receiving this pair, the receiving party first decrypts the digital signature, using the public key of the party who has sent the message, to obtain $h(m)$. Next the receiving party computes the hash of the message $h(m)$. This can be done since the hash function h is a publicly known algorithm and m is already available to the receiver. If this computed value of $h(m)$ matches with what has been obtained by decrypting the digital

signature, the document is indeed signed by the sender.

Impact of Hash Function Collision on a Digital Signature Scheme. Suppose a hash function h is not collision resistant. Then it is feasible to generate two messages M and M' both hashing to the same value. Let $h(M) = h(M') = h$ and the signer's secret key be sk. Since the digital signature is computed on the hash of the message, rather than on the message itself, the pair $(M, E_{sk}(h))$ and $(M', E_{sk}(h))$ will both pass-off as having been signed by the same party. This is called a "*forgery*".

In the case of a forgery, one of the communicating parties may send a message M and sign it, and later claim to have sent the message M'. The other party in the communication can not prove the falsity of this claim.

This shows that it order to be useful, a digital signature scheme must be based on a collision resistant hash function.

History of Cryptographic Hash Functions. Merkle [37] and Damgård [12] independently made the first theoretical study on the construction of hash functions at Crypto '89. Following the design principle proposed in their study, Rivest designed the first widely used hash function MD4 in 1990. The MD4 hash function is described in [42] and [43]. The design of MD4 has inspired a host of other hash functions which were designed later. Following the indications of some weaknesses in MD4, Rivest designed a stronger (but somewhat slower) hash function MD5 in 1992 as a replacement for MD4 [44]. MD4 was indeed found to be weak later, when Dobbertin announced collisions for it in 1994.

The European project RACE Integrity Primitives Evaluation, known by its acronym RIPE, developed a substitute for MD4 in 1995 independently. This proposal was named RIPEMD [7]. The design of RIPEMD was inspired by extended-MD4, a variant of MD4 which did not gain much attention although it was described by Rivest in the original design proposal of MD4 itself. Following cryptanalytic results on MD4, MD5 and RIPEMD, Dobbertin et al. [19] proposed a family of hash functions. This family consisted of four hash function proposals- RIPEMD-128, RIPEMD-160, RIPEMD-256 and RIPEMD-320. RIPEMD-128 was intended to be a replacement for the original RIPEMD proposal with its digest having the same size as RIPEMD, whereas the other three variants had longer digest sizes.

The US governmental body National Institute for Standards and Technology (NIST) proposed a new hash function in 1993. This proposal was named the "Secure Hash Algorithm" or SHA [57]. An year later, NIST announced that there was a "design flaw" in SHA and they announced a minor modification in the algorithm [58]. No details about the "flaw" were provided by NIST. In order to differentiate the two versions of the Secure Hash Algorithm, the original proposal is referred to in the literature as SHA-0 and

the later version as SHA-1. Both the SHA-0 and SHA-1 designs are inspired by the MD4 construction.

In 2002, NIST announced the design of three new hash functions SHA-256, SHA-384 and SHA-512. Later a fourth hash function SHA-224 was also added to these [59]. The design of all these four hash functions is very similar, and hence they are commonly referred to as the SHA-2 hash family. The SHA-2 family takes many design ideas from SHA-0/1. As already mentioned, the SHA-0/1 hash functions themselves are based on MD4. Therefore all the hash functions discussed above are known as the "MD Hash Family" [14].

Apart from the above mentioned hash functions, many other hash designs have been proposed in the literature. Some of these are HAVAL [71], PANAMA [9], Whirlpool [40], TIGER [1] etc.

A Brief History of Attacks on the MD Family of Hash Functions. Attacking a hash function means attacking one of the three main cryptographic properties, the Collision resistance, the Pre-image resistance or the Second pre-image resistance. As already explained, if an attack against a hash function succeeds with an effort lesser than the generic attack, then that hash function is said to be broken.

It often takes long time and deep understanding of the design of a hash function to find shortcut attacks breaking that hash function. Therefore, it is common in literature to study variants of a design and attack these variants instead. As the knowledge of the cryptanalysis community grows with dissemination of the analysis, either the hash function in question is broken in the long run or the confidence of the users grows with respect to its security.

MD4 is the first widely used hash function and hence also the first to receive the attention of cryptanalysis community. Boer and Bosselaers [15] studied a variant of MD4 by considering the last two rounds of MD4 and reported collisions better than the birthday attack against this variant. Vaudeney [62] studied a MD4 variant by considering only the first two rounds of MD4 and reported that this variant was insecure against collision attacks. Dobbertin [18] studied the first two rounds of MD4 and showed that it was not a one way function. However, the major breakthrough in the cryptanalysis of MD4 was achieved by Dobbertin [17] when collisions for the full MD4 were reported. The reported messages were not only colliding for MD4 but also meaningful. In an important paper at Eurocrypt '05, Wang et al. [64] showed a much faster attack against MD4. At the same conference, Wang et al. [66] also presented collision attacks against MD5 and some other hash functions. The attacks of Wang and her team form the basis of many more attacks which have improved the complexities of the MD4 and MD5 attacks.

As already mentioned, the hash function SHA-0 was withdrawn (and replaced by SHA-1) without any tangible justification provided by NIST.

Chabaud and Joux [5] studied SHA-0 and showed that collisions better than the generic attacks could be found against it, thereby providing the first public justification for the decision of the NIST. In this work, the authors showed that SHA-1 was secure against their attack. In another two papers at Crypto '05, Wang et al. [65, 67] showed an improved collision attack against SHA-0 and the first collision attack against the full SHA-1.

The SHA-2 family is an important target for cryptanalysis at this moment. At the time of writing of this thesis, attacks against up to 24-round SHA-2 family are known. The attacks described in this thesis are the most efficient till date.

Known Results for the SHA-2 Family. Gilbert and Handschuh [22] first studied the compression function of SHA-2 family and described the first linearized local collision[2] for SHA-2 family. Mendel et al. [34] used the Gilbert-Handschuh local collision and reported a message pair colliding for 18-round SHA-256 (corrected later in [35]). We studied the linearized local collisions for SHA-2 family in a general framework and obtained 16 new local collisions [47]. Utilizing one of these new local collisions, we obtained colliding message pairs against 18-round SHA-256 [48].

Nikolić and Biryukov [38] started the analysis of SHA-2 using nonlinear differentials and attacked up to 21-round SHA-256. They presented a new local collision in this work which we refer to as the NB local collision. We developed the work of Nikolić and Biryukov (NB) and generalized their technique. By studying the nonlinear local collisions in a combinatorial framework, we obtained another local collision which has certain advantages over the NB local collision. In order to distinguish it from the earlier local collisions, we refer to it as the SS local collision. Also, we extended the number of rounds that can be attacked to 24 for all members of the SHA-2 family. Indesteege et al. [26] attacked up to 24-round SHA-2 members using the NB local collision.

Apart from the above mentioned collision attacks, non-random properties (pseudo collisions, near collisions, pseudo near collisions or differential paths for these) for SHA-2 family or variants of SHA-2 family have also been studied by researchers. We now briefly describe the works in this area. Mendel et al. [34] provided (i) a differential path for 19-round near collision for SHA-256 and (ii) a pseudo near collision for 22-round SHA-256. An earlier work by Matusiewicz et al. [33] studied a very simplified variant of SHA-256 in which the functions Σ_0, Σ_1, σ_0 and σ_1 were removed. Using the Chabaud-Joux disturbance vector approach [5], it was shown in [33] that collisions much faster than brute force search can be found for this variant. The SHA-256

[2]Technical terms, e.g. linearized and non-linear local collisions, pseudo collisions etc., used in the present discussion are formally defined in Chapter 1 of the thesis.

Table 1: Summary of collision attacks against reduced SHA-2 family. Effort is expressed as either the probability of success or as the number of calls to the respective reduced round hash function.

Work	Hash Function	Rounds	Effort		Local Collision	Attack Type	Example
			Prob.	Calls	utilized		provided
[34, 35]	SHA-256	18		*	GH [22]	Linear	yes
[48]	SHA-256	18		**	SS_5 [47]	"	yes
[38]	SHA-256	20	$\frac{1}{3}$		NB [38]	Non-linear	yes
		21	2^{-19}		"	"	yes
[51]	SHA-256/SHA-512	18,20	1	1	SS [51]	"	yes
	SHA-256	21	2^{-15}		"	"	yes
[49]	SHA-256/SHA-512	21	1	1	"	"	yes
[26]	SHA-256	23		2^{18}	NB [38]	"	yes
		24		$2^{28.5}$	"	"	yes
	SHA-512	23		$2^{44.9}$	"	"	yes
		24		2^{53}	"	"	**no**
[50]	SHA-256/SHA-512	22	1	1	SS [51]	"	yes
	SHA-256	23		$2^{11.5}$	"	"	yes
		24		$2^{28.5}$	"	"	yes
		24		$2^{15.5}$ †	"	"	no
	SHA-512	23		$2^{16.5}$	"	"	yes
		24		$2^{32.5}$	"	"	**yes**
		24		$2^{22.5}$ ‡	"	"	no

* It is mentioned in [34, 35] that the effort is 2^0 but no details are provided.
** Effort is given as running a C-program for about 30–40 minutes on a standard PC.
† A table containing 2^{32} entries, each entry of size 8 bytes, is required.
‡ A table containing 2^{64} entries, each entry of size 16 bytes, is required.

hash function with all modular additions replaced by XOR was studied in [69] and it was shown that pseudo collisions for this variant of the hash function can be found much faster than brute force for up to 34 rounds. Nikolić and Biryukov [38] obtained semi-free start collisions for 23-round SHA-256 and semi-free start near collision for 25-round SHA-256. Indesteege et al. [26] have also studied non-random properties of SHA-256 obtaining semi-free start near collision for 31-round SHA-256. The work [26] also reports other non-random properties for smaller number of rounds for SHA-256. Yu et al. [70] have recently reported non random properties for 39-round SHA-256.

Recent Collision Attacks on Reduced Round SHA-2. The work [26] and its different versions [27] (later published as [26]) was done independently and in parallel to ours [46, 49–51]. This work used the NB local collision.

A summary of results on collision attacks against reduced SHA-2 family is given in Table 1.

Organization of this Thesis. Chapter 1 describes the technical preliminaries on Hash functions and differential collision attacks including definitions of various terms used in later chapters. The structure of Merkle-Damgård construction and the design of the SHA-2 hash family is described in the same chapter.

Chapter 2 studies linearized local collisions for the SHA-2 hash family. To obtain these local collisions, we approximate the two nonlinear functions present in the state update step of SHA-2 family by some suitable linear combination of their inputs. This approximation is done in a generic framework which allows us to obtain 16 new local collisions for the SHA-2 family. This work is motivated by the fact that the only prior known local collision for the SHA-2 family was obtained by removing (that is approximating with zero) the two nonlinear functions in the state update step of SHA-2 family [22]. This chapter is based on [47].

Chapter 3 utilizes one of the local collision described in the previous chapter to attack 18-step SHA-256. We describe an algorithm to generate colliding message pairs for 18-step SHA-256 and exhibit such a message pair. The algorithm uses message modification technique to obtain the colliding message pair. The effect of neutral bits on the collision search is also studied briefly in this chapter. Using coding theoretic techniques in a novel way, we generate longer round differential paths for SHA-256. This chapter is based on [48].

Chapter 4 studies non-linear local collisions for the SHA-2 family in a combinatorial framework. This allows us to explain the local collision obtained by Nikolić and Biryukov (NB) in their important work presented at FSE '08 [38]. The combinatorial framework also allows us to generate a different non-linear local collision. We show that the NB local collision can be utilized to obtain deterministic attacks against up to 21-step SHA-2 family, and our new local collision can be used to obtain deterministic attacks against up to 22-step SHA-2 family. We utilize the new local collision to describe attacks against 23 and 24 step SHA-2 family. These attacks have the best complexity at the time of writing this thesis, improving the results of Indesteege et al. at SAC '08 [26]. The improved complexity allows us to obtain the first colliding message pair for 24-step SHA-512. Finally we study some mathematical properties of the NB local collision. This chapter is based on [51], [49], [50] and [52].

Note that our attacks in this chapter (based on the non-linear local collisions) are better than those described in the previous chapter (based on the linearized local collisions). However, the non-linear techniques may be difficult to extend to longer number of rounds. The analysis of linear attacks that we have done in the previous chapter may be useful for such purposes.

Chapter 5 describes the construction of a new hash family obtained by modifying the SHA-2 family. The modifications proposed are minor in nature and do not affect the SHA-2 speeds substantially. On the other hand, the security margin of the new design is expected to be considerably more. In particular, all the known attacks against (reduced) SHA-2 fail against the new design. We describe the justification of the proposed modifications, provide pseudo-code of the implementation of new design and present software

speed comparisons with SHA-2. This chapter is based on [53].

Chapter 6 concludes this thesis with a brief overview and discussing possible directions of further research.

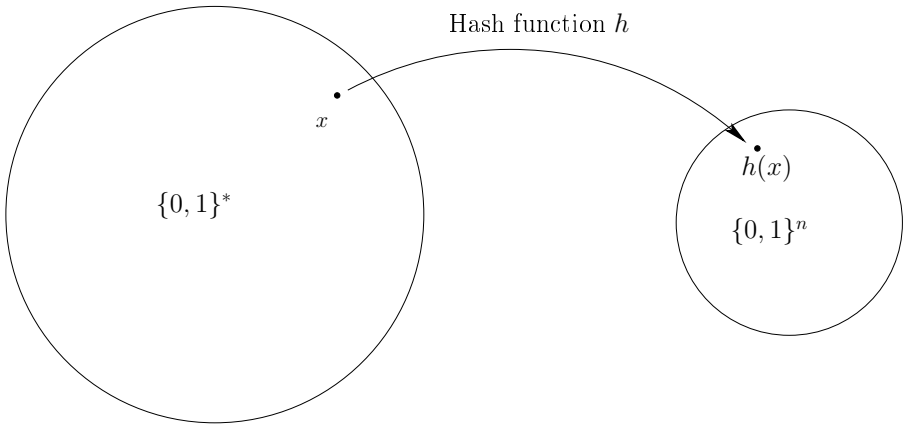
Chapter 1

Preliminaries

1.1 Hash Functions

Hash functions are an important cryptographic primitive. A hash function produces a fixed length digest for an arbitrary sized message [36]. The basic concept of a hash function is shown in Figure 1.1.

Figure 1.1: A hash function produces a fixed length digest for an arbitrary length message.



It is easy to design a function which produces a fixed length digest for a message of an arbitrary length. A naive example is to take first n bits of the message, when the message length is at least n, and discard the rest to produce a digest of length n bits. When the message length is less than n, some null characters can be padded at the end of the message to generate a digest of length n. It is not difficult to construct many more such simple designs for hash functions. However, to be useful for cryptographic applications, a hash function must satisfy certain security requirements. We briefly

describe some of the security requirements for such a function in the next section. For further details, refer to [36, 61].

1.2 Basic Definitions

Definition 1.2.1. ***Hash Function:*** *A hash function is a function $h : X \rightarrow Y$, where $X = \{0,1\}^*$ and $Y = \{0,1\}^n$ for some $n \geq 1$. The output digest $h(x)$ is referred to as the "hash value" or the "digest" of x under h.*

Remark 1.2.2. *Hash functions with infinite domain, along with other desired properties as explained later, are not easy to realize. Therefore, the domain of practical hash functions is bounded above by a very large number, e.g. 2^{64} bits. In practice, however, this puts no restrictions on the use of the hash function. In the rest of this thesis, we will use this restricted definition of a hash function only, i.e. a hash function which has finite size domain.*

The above definition encompasses all types of hash functions, namely the **Cryptographic hash functions** and the **Non-cryptographic hash functions**. Both types of hash functions share some common requirements, e.g. they must be very efficiently computable. However, there are many differences in their design and use. In this thesis, we are only concerned with the former variety. For the latter type of hash functions, see e.g. [30]. In the rest of the thesis, a hash function refers to cryptographic hash function, unless explicitly stated otherwise.

A cryptographic hash function must satisfy certain additional properties for being useful. Roughly speaking, a cryptographic hash function must produce a digest which is random looking and does not have any correlation with its input. Further, a cryptographic hash function is "almost injective" for any randomly selected subset from the domain, having cardinality less than or equal to the co-domain. That is, either all the members from this subset will produce different digests or very few members will produce the same digest. In addition, as already stated for every type of hash function, the evaluation of the digest $h(x)$ from any given x must be very efficient.

All these requirements are difficult to achieve in practice. Therefore, a cryptographic hash function must satisfy some or more of the following properties, depending on the intended use of that hash function [36].

Definition 1.2.3. ***Pre-image Resistance:*** *A hash function $h(.)$ is pre-image resistant if it is computationally infeasible to find an x for any given y such that $h(x) = y$.*

Definition 1.2.4. ***Second Pre-image Resistance:*** *A hash function $h(.)$ is second pre-image resistant if it is computationally infeasible to find an x_2 given any x_1 such that $h(x_1) = h(x_2)$ and $x_1 \neq x_2$.*

Definition 1.2.5. ***Collision Resistance:*** *A hash function $h(.)$ is collision resistant if it is computationally infeasible to find a pair (x_1, x_2), $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.*

By "computational infeasibility", we mean that the complexity of an algorithm to break any one of the security properties is less than the generic attack for breaking that property. For a hash function producing an n-bit digest, the complexity of birthday attack is $2^{n/2}$, and that for the pre-image and the second pre-image attack is 2^n. If an attack can be described against any one of these properties and that attack has better complexity than these generic attacks then it is known as a "breaking" of the hash function.

Remark 1.2.6. *Some other terms are also used in the literature for the above mentioned security properties. These are: pre-image resistant $\equiv$ one-way; second pre-image resistant $\equiv$ weak collision resistant; collision resistance $\equiv$ strong collision resistant ([36], Note 9.2).*

1.3 The Design of Iterated Hash Functions

In order to be practically useful, the hash function is required to be computationally very efficient. Since the message to be hashed can be of arbitrary size, most modern hash functions use an iterative structure to evaluate the hash digest. The message M is divided into multiple block $M_1, M_2, \ldots, M_k$ of equal size, called block size. If the message is not a multiple of block size, a padding scheme is used to increase its length and make it a multiple of block size.

The hash function $h(.)$ uses a compression function $f(.,.)$ on each of the message blocks in turn. The compression function f is designed to compute the digest of a single block message in one iteration. The hash evaluation of M starts with a constant value called IV and produces a digest h_1 corresponding to M_1. The value h_1 is used in place of IV for the next iteration corresponding to M_2 to obtain the next digest h_2. Repeating this process, h_k, the last digest corresponding to M_k is obtained. A final processing step is used to output the hash digest of M as $g(h_k)$, where g is another function. Next we formally define these notions.

1.3.1 Compression Function

Definition 1.3.1. ***Compression function:*** *A function $f : X \times Y \rightarrow Z$ is a compression function if $X = \{0,1\}^m$, $Y = \{0,1\}^l$ and $Z = \{0,1\}^m$, where $l \geq 1$. That is, it compresses a message of length $(m + l)$ into an output of length m.*

Some Desirable Properties of Compression Function

In addition to the three security properties defined earlier, cryptographic hash functions are sometimes also required to satisfy a few additional properties. Some such properties are discussed next. In the following discussion, $f(\mathsf{IV}, x)$ is the compression function used in the design of the hash function h. The function f takes the input IV and the message x. The IV is usually a constant defined by the design of the hash function. We write IV_{std} for the standard IV and IV_1, IV_2, ... for arbitrary IV's.

1. *Near Collision Resistance:* It is computationally difficult to obtain a message pair (x_1,x_2) such that $f(\mathsf{IV}_{std}, x_1)$ and $f(\mathsf{IV}_{std}, x_2)$ (i.e. $h(x_1)$ and $h(x_2)$) differ in very few bits.

2. *Pseudo Collision Resistance:* It is computationally difficult to obtain a message pair (x_1,x_2) for some (IV_1,IV_2) such that $f(\mathsf{IV}_1, x_1) = f(\mathsf{IV}_2, x_2)$.

3. *Pseudo Near Collision Resistance:* It is computationally difficult to obtain a message pair (x_1,x_2) for some (IV_1,IV_2) such that $f(\mathsf{IV}_1, x_1)$ and $f(\mathsf{IV}_2, x_2)$ differ in very few bits.

Remark 1.3.2. *If there exist* IV *and message pairs* (IV_1, x_1), (IV_2, x_2) *satisfying the relation* $f(\mathsf{IV}_1, x_1) = f(\mathsf{IV}_2, x_2)$ *then the messages* x_1 *and* x_2 *are said to be forming a pseudo collision. A further distinction can be made between the cases when one of the two* IV*'s are chosen to be the standard* IV *and when both the* IV*'s are not equal to the standard* IV*. These two cases are known as Semi-free start collisions and Free start collisions respectively.*

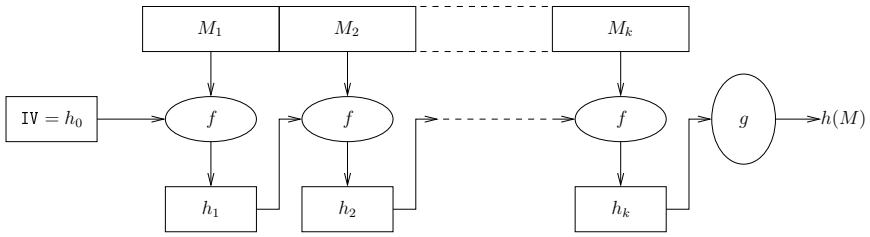
1.3.2 Merkle-Damgård Design Principle

The domain of a compression function can be extended by recursively using it on subparts of the input. The following design is used in most of the modern hash functions.

Definition 1.3.3. ***Merkle-Damgård Design Principle:*** *Let* $f : \{0,1\}^m \times \{0,1\}^l \rightarrow \{0,1\}^m$ *be a compression function as defined above,* IV *be a constant* m*-bit number and* $g : \{0,1\}^m \rightarrow \{0,1\}^m$ *be an additional transformation function. The hash output* $h(M)$ *of a message* $M = M_1||M_2||\ldots||M_k$*, where each* $M_i \in \{0,1\}^l$*, is defined as follows.*

$$\begin{aligned} h_0 &:= \mathsf{IV} \\ h_i &= f(h_{i-1}, M_i) \quad \textit{for } 1 \leq i \leq k \\ h(M) &= g(h_k) \end{aligned}$$

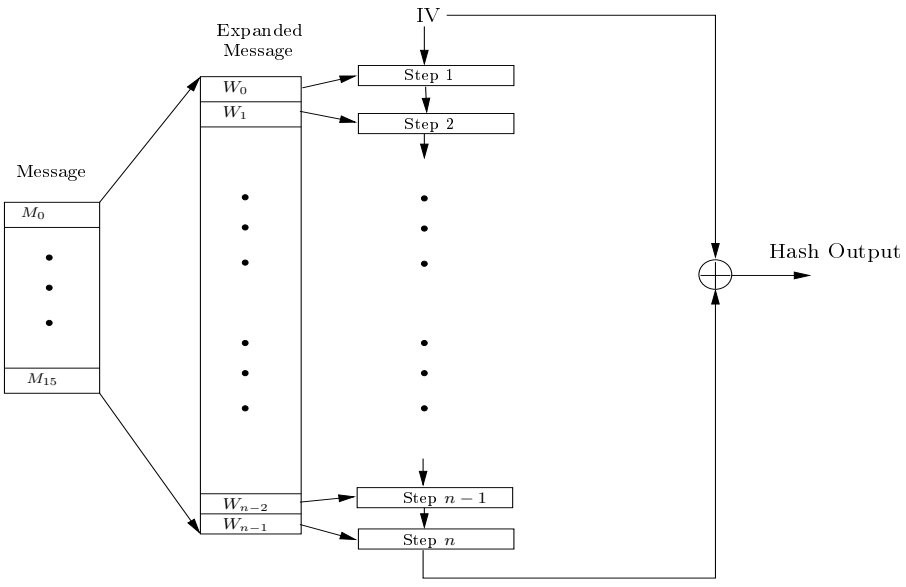
Figure 1.2: The Merkle-Damgård design principle for hash functions.



This design principle is explained in Figure 1.2.

For a single block message, typical hash function compression takes places as in Figure 1.3.

Figure 1.3: Design of an iterated hash function. The schema below represents the hash output of a single block of message. For multi-block messages, the IV is replaced by the hash output of the previous block.



1.3.3 Padding Scheme

Most of the real life hash functions use the iterative structure defined above. The only minor difference is that the padding scheme is always used, irrespective of whether the message length is a multiple of block length or not. The padding scheme has a strong influence on the usability of the hash function.

It is shown in [36, 61] that certain padding schemes lead to security vulnerabilities in the hash function. Such insecure padding schemes are referred to as the "weak padding schemes".

One requirement of the padding scheme is that it should be unambiguous. Let the padding corresponding to the binary string x be denoted by $\mathsf{pad}(x)$ and let x and x' be two binary strings. Then the unambiguousness of the padding scheme means the following.

$$(x||\mathsf{pad}(x) = x'||\mathsf{pad}(x')) \implies (x = x').$$

MD Strengthening: A padding scheme, proposed by Merkle [37] and Damgård [12] at Crypto '89 independently, is to append some block which contains the binary representation of the message length at the end of the message.

Remark 1.3.4. *This requires that the length of the message can be encoded in a single block. It is possible to overcome this restriction and design a padding scheme for messages of infinite length. However, since in this thesis we are considering hash functions with finite domains only, the condition on message length is not restrictive.*

This padding scheme is not only unambiguous, but makes the hash function using it secure. This is due to the following theorem.

Theorem 1.3.5. ***Merkle-Damgård Theorem:*** *Let h be a hash function which utilizes f as a compression function in the Merkle-Damgård structure, with MD strengthening. Then the following result holds.*

$$f \textit{ is pseudo collision resistant} \implies h \textit{ is collision resistant.}$$

Proof. Assume that h is not collision resistant, i.e. we have two binary strings x and x' such that $x \neq x'$ but $h(x) = h(x')$. Let $x = x_1||x_2||x_3||\dots||x_l$ and $x' = x'_1||x'_2||x'_3||\dots||x'_k$, where each of the x_i and x'_j are of the same length which is the size of the second argument of the compression function $f(IV, .)$. Further, let the intermediate chaining hash values for two evaluations of the hash digests for messages x and x' be denoted by h_i and h'_j respectively, with suitable indices i and j.

We consider two cases on the lengths of x and x' as follows.

Case 1: The lengths of x and x' are different.
If the lengths of the two messages are different, then their last blocks are necessarily different. This implies that there exists a pseudo collision for the compression function. This pseudo collision can be explained as follows.

$$h(x) = f(h_{l-1}, x_l) = f(h'_{k-1}, x'_k) = h(x').$$

Case 2: The lengths of x and x' are same.
In this case, we consider two sub cases.

1. If all the intermediate hash values are equal, i.e. $h_1 = h_2$ for all $i \leq l$ with some $x_j \neq x'_j$. In this case, the pseudo collision is obtained as follows.
$$h_j = f(h_{j-1}, x_j) = f(h'_{j-1}, x'_j) = h'_j.$$

2. If some intermediate hash values are unequal, i.e. $h_i \neq h'_i$ for some $i \leq l$ then take the maximal such i and then the pseudo collision is obtained as follows.
$$h_{i+1} = f(h_i, x_{i+1}) = f(h'_i, x'_{i+1}) = h'_{i+1}.$$

■

The theorem implies that if h is not collision resistant then f is not pseudo collision resistant.

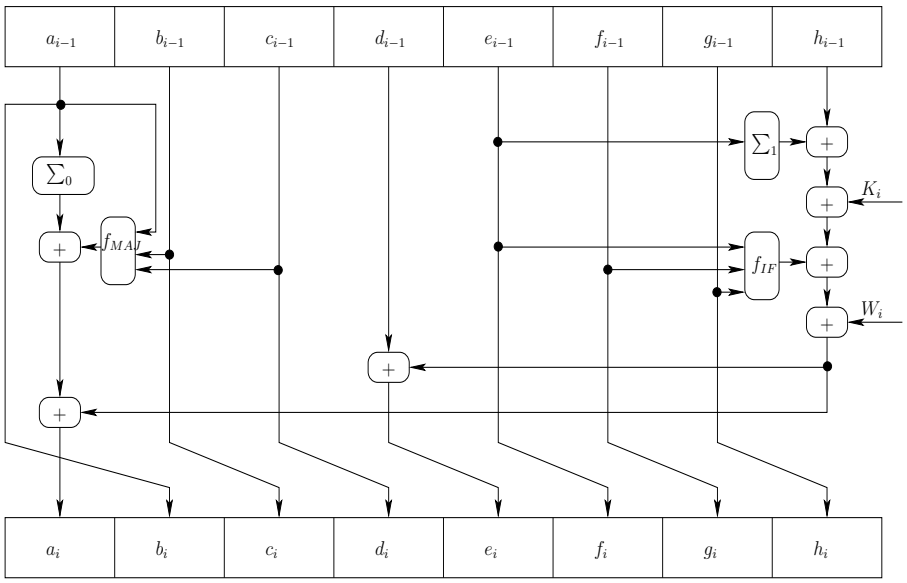
1.4 The SHA-2 Hash Family

The newest members of SHA family of hash functions were standardized by US NIST in 2002 [59]. There are 2 differently designed functions in this standard: the SHA-256 and SHA-512. In addition, the standard also specifies their truncated versions as SHA-224 and SHA-384. The truncation is done by taking only the initial bits of the hash value. The number in the name of the hash function refers to the length of message digest produced by that function.

In this work we are interested in reduced round collision attacks against SHA-256 and SHA-512. Since SHA-224 is the truncated version of SHA-256 and SHA-384 is the truncated version of SHA-512, all the reduced round attack against SHA-256 are also valid for SHA-224 and similarly all the reduced round attacks against SHA-512 are also valid for SHA-384. Next we briefly describe SHA-256 and SHA-512. For details refer to [59].

The round function of SHA-2 hash family is shown in Figure 1.4. Eight registers are used in the evaluation of SHA-2. The initial value in the registers is specified by an $8 \times n$ bit IV, where n is 32 for SHA-256 and 64 for SHA-512. In Round i, the 8 registers are updated from (a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, e_{i-1}, f_{i-1},

Figure 1.4: Round function of SHA-2 hash family



g_{i-1}, h_{i-1}) to (a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i) according to the following equations:

$$\left.\begin{array}{rcl} a_i & = & \Sigma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Sigma_1(e_{i-1}) \\ & & +f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i + W_i \\ b_i & = & a_{i-1} \\ c_i & = & b_{i-1} \\ d_i & = & c_{i-1} \\ e_i & = & d_{i-1} + \Sigma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) \\ & & +h_{i-1} + K_i + W_i \\ f_i & = & e_{i-1} \\ g_i & = & f_{i-1} \\ h_i & = & g_{i-1} \end{array}\right\} \quad (1.4.1)$$

The f_{IF} and the f_{MAJ} are three variable boolean functions defined as:

$$\begin{array}{rcl} f_{IF}(x,y,z) & = & (x \wedge y) \oplus (\neg x \wedge z) \\ f_{MAJ}(x,y,z) & = & (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x) \end{array}$$

The functions Σ_0 and Σ_1 are defined differently for SHA-256 and SHA-512. For SHA-256, these functions are defined as:

$$\begin{array}{rcl} \Sigma_0(x) & = & ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x), \\ \Sigma_1(x) & = & ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x). \end{array}$$

For SHA-512, they are defined as:

$$\begin{array}{rcl} \Sigma_0(x) & = & ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x), \\ \Sigma_1(x) & = & ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x). \end{array}$$

Round i uses a n bit word W_i which is derived from the message and a constant word K_i. The number of rounds r is 64 for SHA-256 and 80 for SHA-512. The hash function operates on a 512 bit/1024 bit message specified as 16 words of 32/64 bits. Given the message words m_0, m_1, $\ldots m_{15}$, the W_i 's are computed using the equation:

$$W_i = \begin{cases} m_i & \text{for } 0 \leq i \leq 15 \\ \sigma_1(m_{i-2}) + m_{i-7} + \sigma_0(m_{i-15}) + m_{i-16} & \text{for } 16 \leq i \leq r \end{cases} \quad (1.4.2)$$

For SHA-256, the functions σ_0 and σ_1 are defined as:

$$\begin{aligned} \sigma_0(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x), \\ \sigma_1(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x). \end{aligned}$$

For SHA-512, they are defined as:

$$\begin{aligned} \sigma_0(x) &= ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x), \\ \sigma_1(x) &= ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x). \end{aligned}$$

The $\mathsf{IV} = (a_{-1}, b_{-1}, c_{-1}, d_{-1}, e_{-1}, f_{-1}, g_{-1}, h_{-1})$ is defined as (`6a09e667`, `bb67ae85`, `3c6ef372`, `a54ff53a`, `510e527f`, `9b05688c`, `1f83d9ab`, `5be0cd19`) for SHA-256. Different IV values are specified for SHA-224, SHA-384 and SHA-512. All additions in Equations 1.4.1 and 1.4.2 are modulo 2^n. Since SHA-224 and SHA-384 are just truncated versions of SHA-256 and SHA-512 respectively, we refer to all the four hash functions as SHA-2 family.

The output hash value of a one block (512/1024 bit) message is obtained by chaining the IV with the register values at the end of the final round as per the Merkle-Damgård construction [12, 37]. A similar strategy is used for multi-block messages, where the IV for next block is taken as the hash output of the previous block.

Reduced Round SHA-2. The number of rounds r is fixed by the specification [59]. For the purpose of analysis, one may work with a lower value of r. In this thesis, we will work with r up to 24. Everything else of the compression function, including the feed-forward, remain the same. Actually, we will not have to bother about the feed-forward, since we will be obtaining collisions for several rounds of the round function itself.

1.5 Differential Collision Attacks

The aim of a collision attack is to produce two different messages both of which map to the same hash output. This is done by employing differential attack against the hash function in question. First a suitable difference of messages is found such that a pair of messages having that difference is

likely to collide to the same hash value with high probability. For example, if a given message differential $\{\Delta W_0, \Delta W_1, \ldots \Delta W_{15}\}$ is likely to generate colliding pairs with probability $\frac{1}{2^8}$ then one needs to try roughly 2^8 different pairs $\{W_0, W_1, \ldots, W_{15}\}$ and $\{W'_0, W'_1, \ldots W'_{15}\}$ having the given difference to get a colliding pair of messages.

However, the probability of the specified differential to generate a collision is likely to be very low for most of the practical hash functions. Hence some sophisticated methods are used to search for the right (colliding) pair, rather than generating them at random. Message modification techniques [64, 66] and neutral bit technique [2] are the two methods to find colliding message pairs.

We next describe some terms and techniques used in the differential attack on hash functions.

1.6 Local Collision

A local collision (LC) is a list of consecutive message word differences, which will cancel themselves when run through some rounds of the hash function evaluation. In this thesis we use two types of local collisions, the XOR based linearized LC and the modular addition based non-linear LC. Let a k-round local collision $\mathcal{L}$ be $\{d_1, d_2, \ldots d_k\}$. The difference between the two types of local collisions can be understood as follows.

$\mathcal{L}$ is called a *linearized local collision* when the message pair $\{M_1, M_2, \ldots, M_k\}$ and $\{M_1 \oplus d_1, M_2 \oplus d_2, \ldots, M_k \oplus d_k\}$ collide for k rounds of evaluation of the linearized version of the hash function. This local collision holds with probability 1 for the linearized version of the hash function, but much less probability for the actual hash function.

$\mathcal{L}$ is called a *non-linear local collision* when the message pair $\{M_1, M_2, \ldots, M_k\}$ and $\{M_1 + d_1, M_2 + d_2, \ldots, M_k + d_k\}$ collide for k rounds of evaluation of the actual hash function. This local collision usually holds with significantly higher probability for the hash function in comparison to the linearized local collision.

All the local collisions for the SHA-2 hash family studied in this thesis are of k =9 rounds.

1.6.1 Local Collision for the Linearized Hash Function

To analyze a hash function, it is often advantageous to consider its linearized version. That is, all the non-linear components in the hash function design are first replaced by their linear approximations. Since the resulting design is linear, we have $h(x) \oplus h(x \oplus \Delta x) = h(\Delta x)$ where h is one round of the hash

function evaluation. Therefore we can look at the behavior of the differential of the messages Δx on the linearized hash function rather than considering a pair of messages x and $x \oplus \Delta x$ operated on by a round of linearized hash function, and then taking their differential.

The simplified version of the hash function is then analyzed for a small number of rounds. The technique from [5] is to introduce a small perturbation in the message at the first round and then correct it by suitable differences in the messages in next few rounds. After introducing some message differences for few rounds it is possible to completely cancel the effect of the original perturbation. A sequence of message word differences which cancel their own effect locally for the linearized version of the hash function are said to construct a "*local collision for the linearized hash function*". For the sake of convenience, we also refer to it as the"*linearized local collision*" in this thesis. Note that the hash function is linearized, not the collision. All the message words are considered to be unrestricted during the search for these local collisions. Finally many linearized local collisions are interleaved to obtain (reduced round) colliding pair of messages.

Attacks on the SHA-0 and SHA-1 hash functions use the 5-round linearized local collision obtained by Chabaud and Joux [5].

For the SHA-2 family, a 9-round local collision by Gilbert and Handschuh [22] is known. We describe sixteen new 9-round local collisions in this thesis.

1.6.2 Non-linear Local Collision

There are instances of hash function designs for which linearized local collisions do not allow attacks with high probability. In such situations, one may look for a local collision for the actual rounds of the hash function involving non-linear rounds. Doing so may lead to restrictions which may be difficult to satisfy. However, it turns out that for the SHA-2 design, the non-linear local collisions and attacks based on them have much higher probabilities of success. The first non-linear local collision for SHA-2 was given by Nikolić and Biryukov [38].

In this thesis, we will further develop the important work of Nikolić and Biryukov and obtain another nonlinear local collision for SHA-2 family. As this work will later show, non-linear local collisions have recently been used to attack reduced round SHA-2 family successfully. Note that, unlike the linearized local collisions, the non-linear local collisions cannot be easily superimposed to obtain longer round collisions.

1.7 Differential Path

The introduction of the message differentials causes the internal registers of hash function to differ too. The propagation of the differences in the registers creates a "*differential path*". If the differential path has been obtained for the linearized version of the hash function then it is referred to as the "*L-characteristic*" [34]. It has also been called the "*linearized differential path*". We also use these terms interchangeably with differential path for the linearized hash function.

Let messages $M = \{M_1, M_2, \ldots, M_k\}$ and $M' = \{M'_1, M'_2, \ldots, M'_k\}$ be used in k Rounds of the hash evaluation. The list of differences in the two copies of the registers at each round is called the differential path (DP). If the two copies of the registers match after the k^{th} Round, then such a DP is called a colliding DP. As in the case of the local collision, the DP can also be of two types depending on whether the differences are XOR based or Modular addition based. These two types are termed linear DP and non-linear DP respectively. In this thesis we use both these types of DPs. A particular DP may be referred to as simply DP, without an explicit prefix, when the context makes the type of that DP obvious.

For the members of SHA-2 family, the linearized differential path for k rounds can be completely specified by differences in the 8 registers for these k rounds along with the k message differences corresponding to the same rounds. If the linearized differential path includes rounds in which the message expansion is involved, then the message words for some rounds are computed on the basis of previous message words. In such a case, the message expansion is also linearized. If the register differences are all zero at the last round of the differential path then such a path is said to be a "*colliding differential path*". The differential of the messages for a colliding differential path are called as "*colliding message differential*".

The linearized differential path holds for the linearized version of the hash function with probability 1, but with much less probability for the actual hash function.

It is straightforward to extend the terms defined above for the actual (non-linear) hash function as well. In this later case, we refer to the terms above as "*non-linear differential path*", "*non-linear colliding differential path*" etc. In order to shorten the terms, we often refer to these terms without the prefix "non-linear" or "linearized". It will be clear from the context as to which version is referred to.

1.8 Chabaud-Joux type Disturbance Vector

Disturbance Vector (DV) is a binary sequence having one at those positions where a linearized LC has been started, and zero otherwise. A linearized local collision starts with an initial difference in the message words. If the rest $k-1$ message word pairs follow the chosen k-round LC, then this will lead to a collision for the linearized version of the hash function after k Rounds. It is possible and straightforward to combine multiple linearized LCs to produce a longer round DP. For example, a k-round linearized LC may be started at Round 1 and another at Round 4. The combined effect of these two LCs will be a $k+3$ Round DP. Using the terminology of [5], we say that the DV for this case is 1001. We say that perturbation differences are introduced at Rounds 1 and 4, and are corrected in the other rounds. We will not be using disturbance vectors in this thesis.

1.9 Message Modification Technique

This technique was effectively used by Wang et al. in [64,66] to attack many hash functions including MD4 and MD5. The basic idea of the method is to first translate conditions on the differential path to bit-level conditions on the internal registers at different steps. A condition on the differential path may be satisfied in multiple ways. In such cases the authors use a set of conditions on internal register bits so that there are no conflicting conditions. Once all the bit-level conditions have been generated, the authors utilize the fact that in all the hash functions they attack, a particular register in the i^{th} step, say a_i, has the following relation with the message word in the i^{th} step, say m_i.

$$a_i = G(.) + m_i, \quad \text{where } G(.) \text{ is some function.}$$

The technique starts with some randomly chosen message pair as candidate pair for collision. Naturally, most of the bit conditions will not be satisfied for these messages. The interesting part of the technique is to now solve the above equation in reverse and find m_i *after* the satisfaction of bit-conditions on a_i. This gives a new value of the message word, which is called as the "modified message".

Modification of the message in this way is easy to do for first few rounds, since the message words can be freely chosen in initial rounds. The authors provide more sophisticated methods to satisfy conditions on registers in later rounds by suitably modifying multiple message words from previous rounds. Once all the message modifications have been carried out, a much lesser set of bit conditions remain. These are satisfied probabilistically by starting the message modification on randomly generated pair of messages. For further details refer to [64,66].

1.10 Neutral Bit Technique

This technique was developed by Biham and Chen [2] and utilized for attacking (reduced) SHA-0 and SHA-1. The authors first make a startling observation that there are many message bits which, even if flipped, do not affect bit-level conditions on registers for long number of rounds. Such message bits are termed "neutral bits". The authors take this concept further and find pairs, triples etc of message bits which, if flipped together, do not break-down conditions on the differential path. Once a set of n neutral bits are found, it is easy to generate 2^n message pairs starting from only a single pair, by flipping all combinations of neutral bits. This gives a big pool of messages to start message modifications.

Since it is possible to generate large number of messages starting from just one pair and all the pairs have similar differential behaviour, it is better to use them, rather than randomly choose messages for modification. A near collision for the full SHA-0 was found using this technique in [2]. For complete details of the technique, refer to [2].

1.11 Multicollisions

Multicollisions are an extension of collisions for hash functions. The basic idea is to find many messages (rather than a pair), all of which hash to the same value. A k-way multicollision means finding a set of k messages all of which have the same digest. Multicollisions were considered in [37] with respect to the ideal goal of a hash function as being completely random.

Extending the birthday bound to cover multicollisions, we expect that one will require $2^{\frac{(k-1)n}{k}}$ effort to find k-way multicollisions for an ideal n-bit hash function. However, Joux [28] showed that constructing 2^t collisions for the MD structure requires an effort which is at most t times the effort required for finding the usual (2-way) collision.

The result of Joux shows that the Merkle-Damgård design principle cannot yield an ideal hash function. There have been a number of later works on designing hash functions with resistance to multicollision attack.

Chapter 2

New Linearized Local Collisions for the SHA-2 Hash Family

2.1 Introduction

The starting point for collision attacks on practical hash functions is a local collision. This is a collision for a fixed number of rounds of the compression function. Details about the message expansion are ignored. There are two types of local collisions – the linearized LC and the non-linear LC. In the linearized LC, all nonlinear components of the hash design are approximated by some suitable linear functions. In the non-linear LC, the nonlinear components remain as in the actual hash function, but some simplifications are used in handling them. The approximations/simplifications of non-linear components are a necessity in order to render the analysis of the local collision tractable.

In this and the next chapter, we discuss the linearized LCs only. Therefore in the rest of this chapter, wherever the term LC is used, it refers to the linearized LC only. Once a local collision is obtained, one attempts to find a collision for the full hash function by taking into account the message expansion and the nonlinear behavior of the hash design. For example, Wang et al.'s attack on the SHA-1 hash function [65] uses the local collision obtained by Chabaud and Joux [5]. For details about this approach one may refer to [5].

2.2 Linearized Local Collisions

2.2.1 The Gilbert-Handschuh Local Collision

Gilbert and Handschuh (GH) [22] were the first to study local collisions in the SHA-2 family. They reported a 9-round local collision and estimated

the probability of the differential path to be 2^{-66}. In this thesis we refer to this local collision as the GH local collision. Most of the early works on the SHA-2 family consider only the GH local collision.

As already remarked in the previous chapter, the GH local collision was obtained by approximating both f_{IF} and f_{MAJ} by 0. In this chapter, we show that both the approximations have one conflicting condition each and this can lead to an impossible differential path.

2.2.2 New Local Collisions

Local collisions are found by forming linear approximations of the Boolean functions f_{IF} and f_{MAJ} involved in round function of SHA-2. We make a systematic analysis of the linear approximations of the two Boolean functions. The differential analysis shows that certain kinds of linear approximations give rise to conflicting conditions.

Given any linear approximations for f_{IF} and f_{MAJ}, we describe a step-by-step method for finding a 9-round local collision for the corresponding linearized round function. This method has been applied on all feasible linear approximations. There are four linear approximations each of f_{MAJ} and f_{IF} which do not have any conflicting conditions. These give rise to a total of 16 different linear approximations without any conflicting conditions. We develop all these approximations to obtain 16 new local collisions without any conflicting conditions. Also, we describe four other local collisions which have one conflicting condition for f_{MAJ} and none for f_{IF}.

The probabilities of these local collisions are a little less than the GH local collision. On the other hand, the absence of conflicting conditions possibly make them more suitable for (reduced round) collision search attacks on the SHA-2 family.

Probabilities of all the local collisions are computed. By our calculations we obtain the probability of the GH local collision to be 2^{-42}. We provide detailed analysis of our probability computations.

Remark 2.2.1. *The probability of the GH local collision was estimated to be 2^{-66} in [22]. Our calculations later in this chapter show this to be 2^{-42}. In [24] this probability was computed to be 2^{-39} when using modular differences (as opposed to XOR differences considered here). Mendel et al. [34] remarked that the probability can be higher than 2^{-39} even when considering XOR differences. However, the details of how this estimate was arrived at were not provided in [34]. As later explained in [25], this increase in the probability is obtained by satisfying three conditions in some special ways. The relative probabilities of the different local collisions will remain the same irrespective of which method is applied to compute the probabilities.*

2.3 Differential Properties of Boolean Functions

Let $f(x)$ be a Boolean function on n variables. By Δx we denote the XOR difference in the input of f, i.e., $\Delta x = x \oplus x'$ for two n-bit strings x and x'. The value of Δx can be any 2^n bit string. Given Δx, define $\Delta f = f(x \oplus \Delta x) \oplus f(x)$. The value of Δf is either 0 or 1 but is not uniquely determined by the value of Δx. Assuming that x is uniformly distributed over $\{0,1\}^n$, the value of Δf is 0 or 1 with certain probabilities.

There are two Boolean functions used in SHA-2, namely the f_{IF} and the f_{MAJ}, which are 3-input bit-wise 'If' and the 'Majority' functions respectively. The three inputs to the functions can have XOR differences of 0 or 1. Depending on their positions, the Boolean functions propagate the differences or absorb them. The differential properties are shown in Table 2.1. The first 3 columns in this table are the input differences to the Boolean functions, whose output differences are listed in next 2 columns. An entry of 0 (resp. 1) in a Boolean function column means that Δf is 0 (resp. 1) with probability 1. An entry $(0,1)$ denotes that Δf is 0 with probability half. We will use this table to compute the probabilities of the differential paths that we show later. Note that the differential properties of Boolean function f_{IF} and f_{MAJ} are also considered in [41], for the analysis of SHA-1 round function, but our presentation is different. In particular, while we list conflicting conditions for different linear approximations of f_{IF} and f_{MAJ} in Table 2.2, Table 2 in [41] lists the probability of output bit flipping when input bits are changed as per a selected difference for all linear functions.

Conflicting Conditions: Suppose we approximate $f(x)$ by a linear function $l(x)$. Note that Δx fixes the value of Δl with probability one. Now suppose that for some Δx, the value of Δf is also determined with probability one and that $\Delta f \neq \Delta l$ for this value of Δx. Then the particular value of Δx for which this occurs is said to be a **conflicting condition** for the approximation of f by l. The complete list of conflicting conditions which arise when f_{IF} and f_{MAJ} are approximated by different linear functions is given in Table 2.2.

The probability that $f_{IF}(a,b,c) = 0$ is $1/2$ and the probability that $f_{IF}(a,b,c) = c$ (or b) is $3/4$. This suggests that approximating f_{IF} by c (or b) should be better than approximating f_{IF} by 0. From Table 2.1, the probability that $\Delta f_{IF} = \Delta c$ is $5/8$, whereas the probability for $\Delta f_{IF} = 0$ is still $1/2$. Thus, on an average, the approximation of f_{IF} by c should be better than that by 0 even for a differential analysis.

Table 2.1: Differential properties of f_{IF} and f_{MAJ}. A single 1 (0) in the last 2 columns means that this value holds with probability 1. The entry (0,1) implies that both the values are possible with probability $\frac{1}{2}$ each.

Δa	Δb	Δc	$\Delta f_{IF}(a,b,c)$	$\Delta f_{MAJ}(a,b,c)$
0	0	0	0	0
0	0	1	(0,1)	(0,1)
0	1	0	(0,1)	(0,1)
0	1	1	1	(0,1)
1	0	0	(0,1)	(0,1)
1	0	1	(0,1)	(0,1)
1	1	0	(0,1)	(0,1)
1	1	1	(0,1)	1

Table 2.2: Conflicting conditions for the different linear approximations of $f_{IF}(a,b,c)$ and $f_{MAJ}(a,b,c)$. The entries in the table provide the values of $(\Delta a, \Delta b, \Delta c)$ which are the conflicting conditions for the corresponding approximation.

	0	a	b	c
f_{IF}	$(0,1,1)$	$(0,1,1)$	none	none
f_{MAJ}	$(1,1,1)$	none	none	none
	$a \oplus b$	$a \oplus c$	$b \oplus c$	$a \oplus b \oplus c$
f_{IF}	none	none	$(0,1,1)$	$(0,1,1)$
f_{MAJ}	$(1,1,1)$	$(1,1,1)$	$(1,1,1)$	none

Explanations of two observations on Page 135 of [34]. These observations were made regarding the presence of impossible characteristics in the GH local collision where both f_{IF} and f_{MAJ} are approximated by 0.

1. If there are three consecutive rounds in the differential path, such that Δa is 1 in the same bit position, then the resulting characteristics is impossible.

2. If there are three consecutive rounds in the differential path, such that there is a bit position where Δe is 1 for the first two rounds and 0 for the third round, then the resulting characteristics is impossible.

The first observation is explained by the fact that the condition $(1, 1, 1)$ is a conflicting condition for the approximation of f_{MAJ} by 0. The second observation is explained by the fact that the condition $(0, 1, 1)$ is a conflicting condition for the approximation of f_{IF} by 0. Note that Mendel et al. [34] also explain these observations on the basis of probability of approximations of f_{IF} and F_{MAJ} being 0 in certain cases, without explicitly mentioning the conditions as presented here. We discuss these observations here since they fit with our unified way of considering the conflicting conditions in the two Boolean functions.

2.4 Linear Approximation of SHA-2 Round Function

All additions in the SHA-2 round function are approximated by XOR. There are many possibilities for the linear approximations of f_{IF} and f_{MAJ} functions. A general form of expressing these approximations is the following

$$\left.\begin{array}{rcl} f_{MAJ}(a,b,c) & = & x_1a \oplus x_2b \oplus x_3c \\ f_{IF}(e,f,g) & = & y_1e \oplus y_2f \oplus y_3g \end{array}\right\} \tag{2.4.1}$$

where (x_1, x_2, x_3) and (y_1, y_2, y_3) are 3-bit strings. Thus, the linear approximations are completely specified by these two strings. Let $\Delta\text{reg}_i = (\Delta a_i, \Delta b_i, \Delta c_i, \Delta d_i, \Delta e_i, \Delta f_i, \Delta g_i, \Delta h_i)$. Then the linearized version of the SHA-2 round function can be expressed by an equation of the form

$$(\Delta\text{reg}_i)^t = A(\Delta\text{reg}_{i-1}, \Delta W_i)^t \tag{2.4.2}$$

where $()^t$ denotes transpose and A is a suitable matrix which is constructed depending upon the particular linear approximation being used.

Let Σ_0 and Σ_1 be functions mapping n-bit binary strings to n-bit binary strings. These functions are defined for SHA-2 members as given in Section 1.4. Let $\mathcal{I}$ be the identity function and $\mathcal{O}$ be the zero function on n-bit

binary strings[1]. Let an n-bit all zero string be denoted by $\mathbf{0}$. Then for a n-bit string x, $\mathcal{I}(x) = x$ and $\mathcal{O}(x) = \mathbf{0} = (0, 0, \ldots, 0)$.

Let b be a bit. Define op_b as follows.

$$\begin{aligned} \mathsf{op}_b &= \mathcal{O} \quad \text{if } b = 0 \\ &= \mathcal{I} \quad \text{if } b = 1. \end{aligned}$$

Finally, let $(\mathsf{op}_b \oplus \Sigma_0)$ be another function defined for an n-bit binary string x as follows.

$$(\mathsf{op}_b \oplus \Sigma_0)(x) = \mathsf{op}_b(x) \oplus \Sigma_0(x).$$

Similarly define $(\mathsf{op}_b \oplus \Sigma_1)$.

The form of A in terms of (x_1, x_2, x_3) and (y_1, y_2, y_3) is given by (2.4.3).

$$A = \begin{bmatrix} p_1 & \mathsf{op}_{x_2} & \mathsf{op}_{x_3} & \mathcal{O} & p_2 & \mathsf{op}_{y_2} & \mathsf{op}_{y_3} & \mathcal{I} & \mathcal{I} \\ \mathcal{I} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{I} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & \mathcal{I} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{I} & p_2 & \mathsf{op}_{y_2} & \mathsf{op}_{y_3} & \mathcal{I} & \mathcal{I} \\ \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{I} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{I} & \mathcal{O} & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{O} & \mathcal{I} & \mathcal{O} & \mathcal{O} \end{bmatrix}$$
$$\text{where } p_1 = (\mathsf{op}_{x_1} \oplus \Sigma_0) \text{ and } p_2 = (\mathsf{op}_{y_1} \oplus \Sigma_1). \tag{2.4.3}$$

Note that the elements of the matrix A are taken from the ring of functions over n-bit strings. Not all of these elements are invertible. For example, the functions $\mathcal{I} \oplus \Sigma_0$ and $\mathcal{I} \oplus \Sigma_1$ are not invertible for both SHA-256 as well as for SHA-512 (see Section 5.2 for more details).

The simplest linear approximation of the SHA-2 round function can be obtained by approximating both f_{MAJ} and f_{IF} by the constant function $\mathcal{O}$ (i.e., $(x_1, x_2, x_3) = (0, 0, 0)$ and $(y_1, y_2, y_3) = (0, 0, 0)$) as has been done by Gilbert and Handschuh [22]. These approximations, however, give rise to two conflicting conditions as has been discussed in Section 2.3. There are four linear approximations of f_{IF} which do not have any conflicting conditions. In Table 2.3, we consider the situation where f_{MAJ} is approximated by the zero function and f_{IF} is approximated by five different functions. One of these five approximations is the zero function and the rest four approximations are those for which f_{IF} does not have any conflicting conditions. From Table 2.2, we find that there are 16 possible combinations of linear approximations of f_{MAJ} and f_{IF} which do not have any conflicting conditions. These are listed in Table 2.4.

[1]We thank an anonymous reviewer of the thesis for pointing out errors and inconsistencies in notation in an earlier version of the thesis and helping us improve readability in this Section.

Table 2.3: Linear approximations for $f_{MAJ}(a,b,c)$, $f_{IF}(e,f,g)$ and the corresponding $(x_1,x_2,x_3,y_1,y_2,y_3)$. Case A has been considered by Gilbert-Handschuh. It has one conflicting condition each for both f_{MAJ} and f_{IF}. Cases B to E have one conflicting condition for f_{MAJ} and none for f_{IF}.

Case	$f_{MAJ}(a,b,c)$	$f_{If}(e,f,g)$	(x_1,x_2,x_3)	(y_1,y_2,y_3)
A	0	0	(0,0,0)	(0,0,0)
B	0	g_{i-1}	(0,0,0)	(0,0,1)
C	0	f_{i-1}	(0,0,0)	(0,1,0)
D	0	$e_{i-1} \oplus g_{i-1}$	(0,0,0)	(1,0,1)
E	0	$e_{i-1} \oplus f_{i-1}$	(0,0,0)	(1,1,0)

Table 2.4: Linear approximations for $f_{MAJ}(a,b,c)$ and $f_{IF}(e,f,g)$ and corresponding $(x_1,x_2,x_3,y_1,y_2,y_3)$. These approximations do not have any conflicting conditions for either f_{MAJ} or f_{IF}.

Case	$f_{MAJ}(a,b,c)$	$f_{IF}(e,f,g)$	(x_1,x_2,x_3)	(y_1,y_2,y_3)
1	a	f	(1,0,0)	(0,1,0)
2	a	g	(1,0,0)	(0,0,1)
3	a	$e \oplus f$	(1,0,0)	(1,1,0)
4	a	$e \oplus g$	(1,0,0)	(1,0,1)
5	b	f	(0,1,0)	(0,1,0)
6	b	g	(0,1,0)	(0,0,1)
7	b	$e \oplus f$	(0,1,0)	(1,1,0)
8	b	$e \oplus g$	(0,1,0)	(1,0,1)
9	c	f	(0,0,1)	(0,1,0)
10	c	g	(0,0,1)	(0,0,1)
11	c	$e \oplus f$	(0,0,1)	(1,1,0)
12	c	$e \oplus g$	(0,0,1)	(1,0,1)
13	$a \oplus b \oplus c$	f	(1,1,1)	(0,1,0)
14	$a \oplus b \oplus c$	g	(1,1,1)	(0,0,1)
15	$a \oplus b \oplus c$	$e \oplus f$	(1,1,1)	(1,1,0)
16	$a \oplus b \oplus c$	$e \oplus g$	(1,1,1)	(1,0,1)

2.5 Technique for Finding Local Collisions

We describe the method for finding a local collision spanning k rounds. For the local collision to exist, the difference of registers at the start and at the end must be zero. Besides, the first and the last message differences must not be zero, to make it exactly a k-round collision. The basic idea is to iterate the linear system in the forward direction; equate the register values to 0 after k rounds and then solve the resulting equations. Note that the standard Gaussian method to solve the linear equations can not be used here since not all the elements of the matrix A are invertible. As described earlier, these are elements of the ring of functions from $\{0,1\}^{32}$ to $\{0,1\}^{32}$. The forward iteration is done in the following manner.

1. $\Delta\text{reg}_0 = (0,0,0,0,0,0,0,0)$.
2. For $i = 1$ to k do
3. $\quad (\Delta\text{reg}_i)^t = A(\Delta\text{reg}_{i-1}, \Delta W_i)^t$;
4. end do.

The procedure provides Δreg_k in terms of $\Delta W_1, \ldots, \Delta W_k$. We now have to set $\Delta\text{reg}_k = 0$ and solve for $\Delta W_1, \ldots, \Delta W_k$. Since the expressions for Δreg_k are quite complicated, there does not seem to be any general method for solving these equations. On the other hand, the equations do have a pattern, which we have exploited to obtain solutions. We explain our method for $k = 9$ for Case B of Table 2.3. Similar methods have been applied to the other cases of this table and to the cases described in Table 2.4. All our computations have been carried out using the symbolic computation package Mathematica [68].

2.5.1 Case B of Table 2.3

The actual values of Δreg_9 in this case is given next.

Δa_9	$\Sigma_0^2\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^5\Delta W_1 \oplus \Sigma_0^8\Delta W_1 \oplus \Sigma_0\Sigma_1\Delta W_1 \oplus \Sigma_0^7\Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_0^6\Sigma_1^2\Delta W_1 \oplus \Sigma_0^5\Sigma_1^3\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_0\Sigma_1^4\Delta W_1 \oplus \Sigma_0^4\Sigma_1^4\Delta W_1 \oplus \Sigma_0^3\Sigma_1^5\Delta W_1 \oplus \Sigma_0^2\Sigma_1^6\Delta W_1 \oplus \Sigma_0\Sigma_1^7\Delta W_1 \oplus \Sigma_1^8\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_0^4\Delta W_2 \oplus \Sigma_0^7\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_0^6\Sigma_1\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_0^2\Sigma_1^2\Delta W_2 \oplus \Sigma_0^5\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_0^4\Sigma_1^3\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Sigma_0^3\Sigma_1^4\Delta W_2 \oplus \Sigma_0^2\Sigma_1^5\Delta W_2 \oplus \Sigma_0\Sigma_1^6\Delta W_2 \oplus \Sigma_1^7\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^6\Delta W_3 \oplus \Sigma_0^5\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_0\Sigma_1^2\Delta W_3 \oplus \Sigma_0^4\Sigma_1^2\Delta W_3 \oplus \Sigma_0^3\Sigma_1^3\Delta W_3 \oplus \Sigma_0^2\Sigma_1^4\Delta W_3 \oplus \Sigma_0\Sigma_1^5\Delta W_3 \oplus \Sigma_1^6\Delta W_3 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_0^2\Delta W_4 \oplus \Sigma_0^5\Delta W_4 \oplus \Sigma_1\Delta W_4 \oplus \Sigma_0^4\Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_0^3\Sigma_1^2\Delta W_4 \oplus \Sigma_0^2\Sigma_1^3\Delta W_4 \oplus \Sigma_0\Sigma_1^4\Delta W_4 \oplus \Sigma_1^5\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_0\Delta W_5 \oplus \Sigma_0^4\Delta W_5 \oplus \Sigma_0^3\Sigma_1\Delta W_5 \oplus \Sigma_0^2\Sigma_1^2\Delta W_5 \oplus \Sigma_0\Sigma_1^3\Delta W_5 \oplus \Sigma_1^4\Delta W_5 \oplus \Delta W_6 \oplus \Sigma_0^3\Delta W_6 \oplus \Sigma_0^2\Sigma_1\Delta W_6 \oplus \Sigma_0\Sigma_1^2\Delta W_6 \oplus \Sigma_1^3\Delta W_6 \oplus \Sigma_0^2\Delta W_7 \oplus \Sigma_0\Sigma_1\Delta W_7 \oplus \Sigma_1^2\Delta W_7 \oplus \Sigma_0\Delta W_8 \oplus \Sigma_1\Delta W_8 \oplus \Delta W_9$

Δb_9	$\Delta W_1 \oplus \Sigma_0\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^7\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_0^6\Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_0^5\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_0^4\Sigma_1^3\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_0^3\Sigma_1^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^5\Delta W_1 \oplus \Sigma_0\Sigma_1^6\Delta W_1 \oplus \Sigma_1^7\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_0^6\Delta W_2 \oplus \Sigma_0^5\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_0^4\Sigma_1^2\Delta W_2 \oplus \Sigma_0^3\Sigma_1^3\Delta W_2 \oplus \Sigma_0^2\Sigma_1^4\Delta W_2 \oplus \Sigma_0\Sigma_1^5\Delta W_2 \oplus \Sigma_1^6\Delta W_2 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_0^5\Delta W_3 \oplus \Sigma_1\Delta W_3 \oplus \Sigma_0^4\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_0^3\Sigma_1^2\Delta W_3 \oplus \Sigma_0^2\Sigma_1^3\Delta W_3 \oplus \Sigma_0\Sigma_1^4\Delta W_3 \oplus \Sigma_1^5\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_0^4\Delta W_4 \oplus \Sigma_0^3\Sigma_1\Delta W_4 \oplus \Sigma_0^2\Sigma_1^2\Delta W_4 \oplus \Sigma_0\Sigma_1^3\Delta W_4 \oplus \Sigma_1^4\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_0^3\Delta W_5 \oplus \Sigma_0^2\Sigma_1\Delta W_5 \oplus \Sigma_0\Sigma_1^2\Delta W_5 \oplus \Sigma_1^3\Delta W_5 \oplus \Sigma_0^2\Delta W_6 \oplus \Sigma_0\Sigma_1\Delta W_6 \oplus \Sigma_1^2\Delta W_6 \oplus \Sigma_0\Delta W_7 \oplus \Sigma_1\Delta W_7 \oplus \Delta W_8$
Δc_9	$\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_0^6\Delta W_1 \oplus \Sigma_0^5\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \oplus \Sigma_0^4\Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^3\Delta W_1 \oplus \Sigma_0^2\Sigma_1^4\Delta W_1 \oplus \Sigma_0\Sigma_1^5\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^5\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0^4\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0^3\Sigma_1^2\Delta W_2 \oplus \Sigma_0^2\Sigma_1^3\Delta W_2 \oplus \Sigma_0\Sigma_1^4\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_0^4\Delta W_3 \oplus \Sigma_0^3\Sigma_1\Delta W_3 \oplus \Sigma_0^2\Sigma_1^2\Delta W_3 \oplus \Sigma_0\Sigma_1^3\Delta W_3 \oplus \Sigma_1^4\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0^3\Delta W_4 \oplus \Sigma_0^2\Sigma_1\Delta W_4 \oplus \Sigma_0\Sigma_1^2\Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_0^2\Delta W_5 \oplus \Sigma_0\Sigma_1\Delta W_5 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_0\Delta W_6 \oplus \Sigma_1\Delta W_6 \oplus \Delta W_7$
Δd_9	$\Sigma_0\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^5\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0^4\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_0^2\Sigma_1^3\Delta W_1 \oplus \Sigma_0\Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^4\Delta W_2 \oplus \Sigma_0^3\Sigma_1\Delta W_2 \oplus \Sigma_0^2\Sigma_1^2\Delta W_2 \oplus \Sigma_0\Sigma_1^3\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^2\Sigma_1\Delta W_3 \oplus \Sigma_0\Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_0^2\Delta W_4 \oplus \Sigma_0\Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_0\Delta W_5 \oplus \Sigma_1\Delta W_5 \oplus \Delta W_6$
Δe_9	$\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_1^8\Delta W_1 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Sigma_1^7\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_1^6\Delta W_3 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_1^5\Delta W_4 \oplus \Sigma_1^4\Delta W_5 \oplus \Delta W_6 \oplus \Sigma_1^3\Delta W_6 \oplus \Sigma_1^2\Delta W_7 \oplus \Sigma_1\Delta W_8 \oplus \Delta W_9$
Δf_9	$\Sigma_0^3\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_1^7\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_1^6\Delta W_2 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1^5\Delta W_3 \oplus \Sigma_1^4\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_1^3\Delta W_5 \oplus \Sigma_1^2\Delta W_6 \oplus \Sigma_1\Delta W_7 \oplus \Delta W_8$
Δg_9	$\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Sigma_1^4\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_1\Delta W_6 \oplus \Delta W_7$
Δh_9	$\Sigma_0\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Sigma_1^4\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_1\Delta W_5 \oplus \Delta W_6$

Below we show how to solve for $\Delta W_1, \ldots, \Delta W_9$ under the condition $\Delta\mathrm{reg}_9 = 0$.

Round 1: The expression for Δh_9 is

$$\begin{aligned}\Delta h_9 &= \Delta W_6 \oplus \Sigma_1(\Delta W_5) \oplus \Sigma_1^2(\Delta W_4) \oplus \Delta W_3 \oplus \Sigma_1^3(\Delta W_3) \oplus \Sigma_1^4(\Delta W_2) \oplus \\ &\quad \Sigma_0(\Delta W_1) \oplus \Sigma_1^2(\Delta W_1) \oplus \Sigma_1^5(\Delta W_1).\end{aligned}$$

Setting $\Delta h_9 = 0$ provides

$$\begin{aligned}\Delta W_6 &= \Sigma_1(\Delta W_5) \oplus \Sigma_1^2(\Delta W_4) \oplus \Delta W_3 \oplus \Sigma_1^3(\Delta W_3) \oplus \Sigma_1^4(\Delta W_2) \oplus \Sigma_0(\Delta W_1) \\ &\quad \oplus \Sigma_1^2(\Delta W_1) \oplus \Sigma_1^5(\Delta W_1). \qquad (2.5.1)\end{aligned}$$

Round 2: Eliminating ΔW_6 from $(\Delta a_9, \ldots, \Delta g_9)$ using (2.5.1), we obtain

$$\begin{aligned}\Delta g_9 &= \Delta W_7 \oplus \Delta W_4 \oplus \Sigma_1(\Delta W_3) \oplus \Sigma_0(\Delta W_2) \oplus \Sigma_1^2(\Delta W_2) \oplus \Delta W_1 \oplus \Sigma_0^2(\Delta W_1) \\ &\quad \oplus \Sigma_0(\Sigma_1(\Delta W_1)) \oplus \Sigma_1^3(\Delta W_1).\end{aligned}$$

Setting $\Delta g_9 = 0$ provides

$$\begin{aligned}\Delta W_7 &= W_4 \oplus \Sigma_1(\Delta W_3) \oplus \Sigma_0(\Delta W_2) \oplus \Sigma_1^2(\Delta W_2) \oplus \Delta W_1 \oplus \Sigma_0^2(\Delta W_1) \\ &\quad \oplus \Sigma_0(\Sigma_1(\Delta W_1)) \oplus \Sigma_1^3(\Delta W_1). \qquad (2.5.2)\end{aligned}$$

Round 3: Eliminating ΔW_7 from $(\Delta a_9, \ldots, \Delta f_9)$ using (2.5.2), we obtain

$$\begin{aligned}\Delta f_9 &= \Delta W_8 \oplus \Delta W_5 \oplus \Sigma_1(\Delta W_4) \oplus \Sigma_0(\Delta W_3) \oplus \Sigma_1^2(\Delta W_3) \oplus \Delta W_2 \oplus \Sigma_0^2(\Delta W_2) \\ &\quad \oplus \Sigma_0(\Sigma_1(\Delta W_2)) \oplus \Sigma_1^3(\Delta W_2) \oplus \Sigma_0^3(\Delta W_1) \oplus \Sigma_0^2(\Sigma_1(\Delta W_1)) \\ &\quad \oplus \Sigma_0(\Sigma_1^2(\Delta W_1)) \oplus \Sigma_1^4(\Delta W_1).\end{aligned}$$

Setting $\Delta f_9 = 0$ provides

$$\begin{aligned}\Delta W_8 &= \Delta W_5 \oplus \Sigma_1(\Delta W_4) \oplus \Sigma_0(\Delta W_3) \oplus \Sigma_1^2(\Delta W_3) \oplus W_2 \oplus \Sigma_0^2(\Delta W_2) \oplus \\ &\quad \Sigma_0(\Sigma_1(\Delta W_2)) \oplus \Sigma_1^3(\Delta W_2) \oplus \Sigma_0^3(\Delta W_1) \oplus \Sigma_0^2(\Sigma_1(\Delta W_1)) \oplus \\ &\quad \Sigma_0(\Sigma_1^2(\Delta W_1)) \oplus \Sigma_1^4(\Delta W_1). \qquad (2.5.3)\end{aligned}$$

Round 4: Eliminating ΔW_8 in $(\Delta a_9, \ldots, \Delta e_9)$ using (2.5.3) we obtain

$$\begin{aligned}\Delta e_9 &= \Delta W_9 \oplus \Sigma_0(\Delta W_4) \oplus \Sigma_0^2(\Delta W_3) \oplus \Sigma_0(\Sigma_1(\Delta W_3)) \oplus \Sigma_0^3(\Delta W_2) \oplus \\ &\quad \Sigma_0^2(\Sigma_1(\Delta W_2)) \oplus \Sigma_0(\Sigma_1^2(\Delta W_2)) \oplus \Delta W_9 \oplus \Sigma_0(\Delta W_1) \oplus \Sigma_0^4(\Delta W_1) \oplus \\ &\quad \Sigma_0^3(\Sigma_1(\Delta W_1)) \oplus \Sigma_0^2(\Sigma_1^2(\Delta W_1)) \oplus \Sigma_0(\Sigma_1^3(\Delta W_1)).\end{aligned}$$

Setting $\Delta e_9 = 0$ provides

$$\begin{aligned}\Delta W_9 &= \Sigma_0(\Delta W_4) \oplus \Sigma_0^2(\Delta W_3) \oplus \Sigma_0(\Sigma_1(\Delta W_3)) \oplus \Sigma_0^3(\Delta W_2) \oplus \Sigma_0^2(\Sigma_1(\Delta W_2)) \\ &\quad \oplus \Sigma_0(\Sigma_1^2(\Delta W_2)) \oplus \Delta W_1 \oplus \Sigma_0(\Delta W_1) \oplus \Sigma_0^4(\Delta W_1) \oplus \Sigma_0^3(\Sigma_1(\Delta W_1)) \\ &\quad \oplus \Sigma_0^2(\Sigma_1^2(\Delta W_1)) \oplus \Sigma_0(\Sigma_1^3(\Delta W_1)). \qquad (2.5.4)\end{aligned}$$

Round 5: Eliminating ΔW_9 in $(\Delta a_9, \ldots, \Delta d_9)$ using (2.5.3) we obtain

$$\begin{aligned}\Delta d_9 &= \Sigma_0(\Delta W_5) \oplus \Sigma_0^2(\Delta W_4) \oplus \Sigma_0(\Sigma_1(\Delta W_4)) \oplus \Sigma_0^3(\Delta W_3) \oplus \Sigma_0^2(\Sigma_1(\Delta W_3)) \oplus \\ &\quad \Sigma_0(\Sigma_1^2(\Delta W_3)) \oplus \Delta W_2 \oplus \Sigma_0(\Delta W_2) \oplus \Sigma_0^4(\Delta W_2) \oplus \Sigma_0^3(\Sigma_1(\Delta W_2)) \oplus \\ &\quad \Sigma_0^2(\Sigma_1^2(\Delta W_2)) \oplus \Sigma_0(\Sigma_1^3(\Delta W_2)) \oplus \Sigma_0^2(\Delta W_1) \oplus \Sigma_0^5(\Delta W_1) \oplus \Sigma_1(\Delta W_1) \oplus \\ &\quad \Sigma_0^4(\Sigma_1(\Delta W_1)) \oplus \Sigma_0^3(\Sigma_1^2(\Delta W_1)) \oplus \Sigma_0^2(\Sigma_1^3(\Delta W_1)) \oplus \Sigma_0(\Sigma_1^4(\Delta W_1)).\end{aligned}$$

Now the situation is different from the previous 4 rounds. In the expression for Δd_9 we do not have any ΔW_i whose "coefficient" is 1. Only ΔW_5 occurs once with a "coefficient" of Σ_0. We solve for ΔW_5 in the following manner. Set

$$\Delta W_2 = \Sigma_0(x) \oplus \Sigma_1(\Delta W_1), \tag{2.5.5}$$

where x is a variable to be determined later.

With this substitution, we have $\Delta d_9 = \Sigma_0(\Delta W_5 \oplus X)$, for some expression X which we provide shortly. Now setting $\Delta d_9 = 0$, provides one solution to be $\Delta W_5 = X$, where the value of X is given by the right side of the following expression.

$$\begin{aligned} \Delta W_5 \quad = \quad & (1 \oplus \Sigma_0 \oplus \Sigma_0^4\Sigma_0^3\Sigma_1 \oplus \Sigma_0^2\Sigma_1^2 \oplus \Sigma_0\Sigma_1^3)(x) \oplus \Sigma_0(\Delta W_4) \oplus \Sigma_1(\Delta W_4) \oplus \\ & \Sigma_0^2(\Delta W_3) \oplus \Sigma_0(\Sigma_1(\Delta W_3)) \oplus \Sigma_1^2(\Delta W_3) \oplus \Sigma_0(\Delta W_1) \oplus \Sigma_0^4(\Delta W_1) \oplus \\ & \Sigma_1(\Delta W_1). \end{aligned} \tag{2.5.6}$$

Round 6: Eliminating ΔW_5 in $(\Delta a_9, \Delta b_9, \Delta c_9)$ using (2.5.6) we obtain

$$\Delta c_9 = \Sigma_0^2(x) \oplus \Sigma_0(\Sigma_1(x)) \oplus \Delta W_3 \oplus \Sigma_0^2(\Delta W_1).$$

Setting $\Delta c_9 = 0$ provides

$$\Delta W_3 \quad = \quad \Sigma_0^2(x) \oplus \Sigma_0(\Sigma_1(x)) \oplus \Sigma_0^2(\Delta W_1). \tag{2.5.7}$$

Round 7: Eliminating ΔW_3 in $(\Delta a_9, \Delta b_9)$ using (2.5.7) we obtain

$$\Delta b_9 = \Sigma_0^2(\Sigma_1(x)) \oplus \Delta W_4 \oplus \Delta W_1 \oplus \Sigma_0^2(\Sigma_1(\Delta W_1)).$$

Setting $\Delta b_9 = 0$, provides

$$\Delta W_4 \quad = \quad \Sigma_0^2(\Sigma_1(x)) \oplus \Delta W_1 \oplus \Sigma_0^2(\Sigma_1(\Delta W_1)). \tag{2.5.8}$$

Round 8: Eliminating ΔW_4 from Δa_9 using (2.5.8), we obtain

$$\Delta a_9 = x \oplus \Delta W_1.$$

Setting $\Delta a_9 = 0$ provides

$$\Delta W_1 \quad = \quad x. \tag{2.5.9}$$

Equations (2.5.1), (2.5.2), (2.5.3), (2.5.4), (2.5.5), (2.5.6), (2.5.7), (2.5.8), and (2.5.9) form a solution to the problem of finding a local collision for the

linearized round function. In this form, the equations are not easy to handle. But, if we start the process of back substitution, i.e., use $\Delta W_1 = x$ in (2.5.8) and then use the values of ΔW_1 and ΔW_4 in (2.5.7) and so on, then the solution is substantially simplified and we finally obtain

$$(\Delta W_1, \ldots, \Delta W_9) \; = \\ (x, \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(\Sigma_1(x)), x, \Sigma_0(x) \oplus x, \Sigma_0(x) \oplus \Sigma_1(x), 0, x, x).$$

2.5.2 A Difficult Example: Case 3 of Table 2.4

The technique described in the previous subsection does not always work. There are cases when we cannot solve the equations in the manner described earlier. A slightly modified method is used for such cases. We briefly describe this procedure for the Case 3 of Table 2.4. First we give the actual values of Δreg_9 for this case.

Δa_9	$\Sigma_0\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_0^6\Delta W_1 \oplus \Sigma_0^7\Delta W_1 \oplus \Sigma_0^8\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_0^3\Sigma_1\Delta W_1 \oplus \Sigma_0^6\Sigma_1\Delta W_1 \oplus \Sigma_0^7\Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_0^4\Sigma_1^2\Delta W_1 \oplus \Sigma_0^5\Sigma_1^2\Delta W_1 \oplus \Sigma_0^6\Sigma_1^2\Delta W_1 \oplus \Sigma_0^4\Sigma_1^3\Delta W_1 \oplus \Sigma_0^5\Sigma_1^3\Delta W_1 \oplus \Sigma_0^2\Sigma_1^4\Delta W_1 \oplus \Sigma_0^3\Sigma_1^4\Delta W_1 \oplus \Sigma_0^4\Sigma_1^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^5\Delta W_1 \oplus \Sigma_0^3\Sigma_1^5\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_0\Sigma_1^6\Delta W_1 \oplus \Sigma_0^2\Sigma_1^6\Delta W_1 \oplus \Sigma_1^7\Delta W_1 \oplus \Sigma_0\Sigma_1^7\Delta W_1 \oplus \Sigma_1^8\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_0^4\Delta W_2 \oplus \Sigma_0^5\Delta W_2 \oplus \Sigma_0^7\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0^6\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0^2\Sigma_1^2\Delta W_2 \oplus \Sigma_0^3\Sigma_1^2\Delta W_2 \oplus \Sigma_0^5\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_0^4\Sigma_1^3\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Sigma_0\Sigma_1^4\Delta W_2 \oplus \Sigma_0^3\Sigma_1^4\Delta W_2 \oplus \Sigma_0^2\Sigma_1^5\Delta W_2 \oplus \Sigma_0\Sigma_1^6\Delta W_2 \oplus \Sigma_1^7\Delta W_2 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^5\Delta W_3 \oplus \Sigma_0^6\Delta W_3 \oplus \Sigma_0^2\Sigma_1\Delta W_3 \oplus \Sigma_0^3\Sigma_1\Delta W_3 \oplus \Sigma_0^4\Sigma_1\Delta W_3 \oplus \Sigma_0^5\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_0\Sigma_1^2\Delta W_3 \oplus \Sigma_0^3\Sigma_1^2\Delta W_3 \oplus \Sigma_0^4\Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_0\Sigma_1^3\Delta W_3 \oplus \Sigma_0^2\Sigma_1^3\Delta W_3 \oplus \Sigma_0^3\Sigma_1^3\Delta W_3 \oplus \Sigma_0\Sigma_1^4\Delta W_3 \oplus \Sigma_0^2\Sigma_1^4\Delta W_3 \oplus \Sigma_1^5\Delta W_3 \oplus \Sigma_0\Sigma_1^5\Delta W_3 \oplus \Sigma_1^6\Delta W_3 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_0^2\Delta W_4 \oplus \Sigma_0^5\Delta W_4 \oplus \Sigma_1\Delta W_4 \oplus \Sigma_0^2\Sigma_1\Delta W_4 \oplus \Sigma_0^4\Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_0^3\Sigma_1^2\Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_0^2\Sigma_1^3\Delta W_4 \oplus \Sigma_0\Sigma_1^4\Delta W_4 \oplus \Sigma_1^5\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_0^2\Delta W_5 \oplus \Sigma_0^3\Delta W_5 \oplus \Sigma_0^4\Delta W_5 \oplus \Sigma_0^2\Sigma_1\Delta W_5 \oplus \Sigma_0^3\Sigma_1\Delta W_5 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_0\Sigma_1^2\Delta W_5 \oplus \Sigma_0^2\Sigma_1^2\Delta W_5 \oplus \Sigma_1^3\Delta W_5 \oplus \Sigma_0\Sigma_1^3\Delta W_5 \oplus \Sigma_1^4\Delta W_5 \oplus \Delta W_6 \oplus \Sigma_0\Delta W_6 \oplus \Sigma_0^3\Delta W_6 \oplus \Sigma_0^2\Sigma_1\Delta W_6 \oplus \Sigma_0\Sigma_1^2\Delta W_6 \oplus \Sigma_1^3\Delta W_6 \oplus \Sigma_0\Delta W_7 \oplus \Sigma_0^2\Delta W_7 \oplus \Sigma_1\Delta W_7 \oplus \Sigma_0\Sigma_1\Delta W_7 \oplus \Sigma_1^2\Delta W_7 \oplus \Sigma_0\Delta W_8 \oplus \Sigma_1\Delta W_8 \oplus \Delta W_9$

Δb_9	$\Sigma_0\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^5\Delta W_1 \oplus \Sigma_0^7\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0^6\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_0^5\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_0^4\Sigma_1^3\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_0\Sigma_1^4\Delta W_1 \oplus \Sigma_0^3\Sigma_1^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^5\Delta W_1 \oplus \Sigma_0\Sigma_1^6\Delta W_1 \oplus \Sigma_1^7\Delta W_1 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_0^5\Delta W_2 \oplus \Sigma_0^6\Delta W_2 \oplus \Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_0^3\Sigma_1\Delta W_2 \oplus \Sigma_0^4\Sigma_1\Delta W_2 \oplus \Sigma_0^5\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_0^3\Sigma_1^2\Delta W_2 \oplus \Sigma_0^4\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_0\Sigma_1^3\Delta W_2 \oplus \Sigma_0^2\Sigma_1^3\Delta W_2 \oplus \Sigma_0^3\Sigma_1^3\Delta W_2 \oplus \Sigma_0\Sigma_1^4\Delta W_2 \oplus \Sigma_0^2\Sigma_1^4\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Sigma_0\Sigma_1^5\Delta W_2 \oplus \Sigma_1^6\Delta W_2 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_0^5\Delta W_3 \oplus \Sigma_1\Delta W_3 \oplus \Sigma_0^2\Sigma_1\Delta W_3 \oplus \Sigma_0^4\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_0^3\Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_0^2\Sigma_1^3\Delta W_3 \oplus \Sigma_0\Sigma_1^4\Delta W_3 \oplus \Sigma_1^5\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0^2\Delta W_4 \oplus \Sigma_0^3\Delta W_4 \oplus \Sigma_0^4\Delta W_4 \oplus \Sigma_0^2\Sigma_1\Delta W_4 \oplus \Sigma_0^3\Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_0\Sigma_1^2\Delta W_4 \oplus \Sigma_0^2\Sigma_1^2\Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_0\Sigma_1^3\Delta W_4 \oplus \Sigma_1^4\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_0\Delta W_5 \oplus \Sigma_0^3\Delta W_5 \oplus \Sigma_0^2\Sigma_1\Delta W_5 \oplus \Sigma_0\Sigma_1^2\Delta W_5 \oplus \Sigma_1^3\Delta W_5 \oplus \Sigma_0\Delta W_6 \oplus \Sigma_0^2\Delta W_6 \oplus \Sigma_1\Delta W_6 \oplus \Sigma_0\Sigma_1\Delta W_6 \oplus \Sigma_1^2\Delta W_6 \oplus \Sigma_0\Delta W_7 \oplus \Sigma_1\Delta W_7 \oplus \Delta W_8$
Δc_9	$\Sigma_0^3\Delta W_1 \oplus \Sigma_0^5\Delta W_1 \oplus \Sigma_0^6\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_0^3\Sigma_1\Delta W_1 \oplus \Sigma_0^4\Sigma_1\Delta W_1 \oplus \Sigma_0^5\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_0^4\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_0\Sigma_1^3\Delta W_1 \oplus \Sigma_0^2\Sigma_1^3\Delta W_1 \oplus \Sigma_0^3\Sigma_1^3\Delta W_1 \oplus \Sigma_0\Sigma_1^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Sigma_0\Sigma_1^5\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^5\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_0^4\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0^3\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_0^2\Sigma_1^3\Delta W_2 \oplus \Sigma_0\Sigma_1^4\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^4\Delta W_3 \oplus \Sigma_0^2\Sigma_1\Delta W_3 \oplus \Sigma_0^3\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_0\Sigma_1^2\Delta W_3 \oplus \Sigma_0^2\Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_0\Sigma_1^3\Delta W_3 \oplus \Sigma_1^4\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_0^3\Delta W_4 \oplus \Sigma_0^2\Sigma_1\Delta W_4 \oplus \Sigma_0\Sigma_1^2\Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_0\Delta W_5 \oplus \Sigma_0^2\Delta W_5 \oplus \Sigma_1\Delta W_5 \oplus \Sigma_0\Sigma_1\Delta W_5 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_0\Delta W_6 \oplus \Sigma_1\Delta W_6 \oplus \Delta W_7$
Δd_9	$\Sigma_0\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^5\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_0^4\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_0^2\Sigma_1^3\Delta W_1 \oplus \Sigma_0\Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_0^4\Delta W_2 \oplus \Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_0^3\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_0^2\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_0\Sigma_1^3\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^2\Sigma_1\Delta W_3 \oplus \Sigma_0\Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_0^2\Delta W_4 \oplus \Sigma_1\Delta W_4 \oplus \Sigma_0\Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_0\Delta W_5 \oplus \Sigma_1\Delta W_5 \oplus \Delta W_6$
Δe_9	$\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_1^8\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Sigma_1^6\Delta W_2 \oplus \Sigma_1^7\Delta W_2 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1^6\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0\Delta W_4 \oplus \Sigma_1^4\Delta W_4 \oplus \Sigma_1^5\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_1^4\Delta W_5 \oplus \Delta W_6 \oplus \Sigma_1\Delta W_6 \oplus \Sigma_1^2\Delta W_6 \oplus \Sigma_1^3\Delta W_6 \oplus \Sigma_1^2\Delta W_7 \oplus \Delta W_8 \oplus \Sigma_1\Delta W_8 \oplus \Delta W_9$
Δf_9	$\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Sigma_1^7\Delta W_1 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_1^6\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_1^4\Delta W_3 \oplus \Sigma_1^5\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_1^4\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_1\Delta W_5 \oplus \Sigma_1^2\Delta W_5 \oplus \Sigma_1^3\Delta W_5 \oplus \Sigma_1^2\Delta W_6 \oplus \Delta W_7 \oplus \Sigma_1\Delta W_7 \oplus \Delta W_8$
Δg_9	$\Sigma_0^2\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_1^6\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Sigma_1^5\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1^4\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_1\Delta W_4 \oplus \Sigma_1^2\Delta W_4 \oplus \Sigma_1^3\Delta W_4 \oplus \Sigma_1^2\Delta W_5 \oplus \Delta W_6 \oplus \Sigma_1\Delta W_6 \oplus \Delta W_7$
Δh_9	$\Delta W_1 \oplus \Sigma_0\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_1^2\Delta W_4 \oplus \Delta W_5 \oplus \Sigma_1\Delta W_5 \oplus \Delta W_6$

Rounds 1 to 4: Using the method described earlier, we can obtain

$$\begin{aligned}\Delta W_9 &= \Sigma_0^2\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_0^4\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_0^3\Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \\ &\oplus\Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_0\Sigma_1^3\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \Sigma_1\Delta W_2 \\ &\oplus\Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_0\Sigma_1^2\Delta W_2 \oplus \Sigma_0\Delta W_3 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_1\Delta W_3 \\ &\oplus\Sigma_0\Sigma_1\Delta W_3 \oplus \Delta W_4 \oplus \Sigma_0\Delta W_4 \qquad (2.5.10)\\ \Delta W_8 &= \Sigma_0\Delta W_1 \oplus \Sigma_0^3\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_0\Sigma_1^2\Delta W_1 \\ &\oplus\Sigma_0\Delta W_2 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_0\Sigma_1\Delta W_2 \oplus \Delta W_3 \oplus \\ &\Sigma_0\Delta W_3 \qquad (2.5.11)\\ \Delta W_7 &= \Delta W_1 \oplus \Sigma_0\Delta W_1 \oplus \Sigma_0^2\Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_0\Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \\ &\oplus\Sigma_1^4\Delta W_1 \oplus \Sigma_0\Delta W_2 \oplus \Sigma_1\Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_1^3\Delta W_2 \oplus \Sigma_1^2\Delta W_3 \\ &\oplus\Delta W_4 \oplus \Sigma_1\Delta W_4 \oplus \Delta W_5 \qquad (2.5.12)\\ \Delta W_6 &= \Delta W_1 \oplus \Sigma_0\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_1^5\Delta W_1 \oplus \Delta W_2 \oplus \Sigma_1^2\Delta W_2 \oplus \Sigma_1^4\Delta W_2 \\ &\oplus\Delta W_3 \oplus \Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1^3\Delta W_3 \oplus \Sigma_1^2\Delta W_4 \oplus \Delta W_5 \oplus \\ &\Sigma_1\Delta W_5 \qquad (2.5.13)\end{aligned}$$

Now in the expression for Δd_9, we do not have any ΔW_i with coefficient "1". Therefore, we let the sum of all the terms which do not have Σ_0 in their coefficients be $\Sigma_0 X$. This substitution results in

$$\begin{aligned}\Delta W_5 &= \Sigma_0 X \oplus \Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_1^4\Delta W_1 \oplus \Sigma_1^3\Delta W_2 \\ &\oplus\Sigma_1\Delta W_3 \oplus \Sigma_1^2\Delta W_3 \oplus \Sigma_1\Delta W_4 \qquad (2.5.14)\end{aligned}$$

where X is a variable whose value is not yet known.

Substituting this expression for ΔW_5 in $\Delta d_9 = 0$, we still get an equation in which none of the variables has a coefficient of Σ_0 only. To get such a variable, we sum all the terms which have no Σ_0 coefficient and equate this to $\Sigma_0 Y$, where Y is another variable. This results in the following substitution

$$\Delta W_4 = \Sigma_0 Y \oplus X \oplus \Delta W_1 \oplus \Sigma_1\Delta W_1 \oplus \Sigma_1^2\Delta W_1 \oplus \Sigma_1^3\Delta W_1 \oplus \Sigma_1^2\Delta W_2 \oplus \Delta W_3 \oplus \Sigma_1\Delta W_3 \qquad (2.5.15)$$

Now we need to solve $\Delta d_9 = 0$. Still the form of this equation is

$$\begin{aligned}\Delta d_9 &= \Sigma_0^5\Delta W_1 \oplus \Sigma_0^4\Sigma_1\Delta W_1 \oplus \Sigma_0^2\Sigma_1^2\Delta W_1 \oplus \Sigma_0^3\Sigma_1^2\Delta W_1 \oplus \Sigma_0^2\Delta W_2 \oplus \Sigma_0^3\Delta W_2 \oplus \\ &\Sigma_0^4\Delta W_2 \oplus \Sigma_0^2\Sigma_1\Delta W_2 \oplus \Sigma_0^3\Sigma_1\Delta W_2 \oplus \Sigma_0^2\Delta W_3 \oplus \Sigma_0^3\Delta W_3 \oplus \Sigma_0^2 Y \oplus \\ &\Sigma_0^3 Y \qquad (2.5.16)\end{aligned}$$

In this expression for Δd_9, we note that the coefficient of ΔW_3 is $\Sigma_0^2(1+\Sigma_0)$. To solve for ΔW_3 we try to generate the same coefficient in other terms too. This can be done if we substitute

$$\Delta W_2 = \Sigma_0\Delta W_1 \oplus c_1\Delta W_1 \oplus (1 \oplus \Sigma_0)Z \qquad (2.5.17)$$

where Z is another variable unknown as of now. With these substitutions, $\Delta d_9 = 0$ gives

$$\Delta W_3 = \Sigma_0 \Delta W_1 \oplus \Sigma_1 \Delta W_1 \oplus \Sigma_0 \Sigma_1 \Delta W_1 \oplus Y \oplus Z \oplus \Sigma_0^2 Z \tag{2.5.18}$$

Now solving for $\Delta c_9 = 0$,$\Delta b_9 = 0$ and $\Delta a_9 = 0$ with these values of $\Delta W_9 \ldots \Delta W_3$ substituted, we get

$$Z = 0 \tag{2.5.19}$$
$$Y = 0 \tag{2.5.20}$$
$$X = 0 \tag{2.5.21}$$

Taking ΔW_1 to be x and then back substituting all the variables results in the solution

$$\begin{aligned}(\Delta W_1, \ldots, \Delta W_9) \quad = \quad & (x, \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x)), \\ & x \oplus \Sigma_0(x), x, \Sigma_0(x) \oplus \Sigma_1(x), x, 0, x).\end{aligned}$$

2.6 Results

The detailed differential paths for the cases of Table 2.3 are shown in Table 2.7. The differential paths for the cases in Table 2.4 are shown in Tables 2.9, A.1, A.2 and A.3. Table 2.9 is provided in this chapter and the other three tables are given in Section A.1 in the Appendix.

2.6.1 Probability of Linear DP

A linear DP holds with probability one for the linearized version of the round function. However, when we move to the actual round function, then it holds with lower probability which in some cases may even be zero. If the differential path holds with probability zero for the actual round function, then we call it to be an impossible differential path. Such impossible differential paths arise due to the conflicting conditions in the approximations of the constituent functions by linear functions. Later we will show examples of such differential paths including one obtained from the Gilbert-Handschuh local collision.

We next describe how we computed the probability for a differential path. This computation is based on the following three points.

1. If a and b be two words which differ in one bit position and x be a random word, then $a + x$ and $b + x$ also differ in one bit position with probability one if the differing bit is the most significant bit, else with probability half. (This was also mentioned in [22].)

2. We also assume that if a and b differ in k different bit positions none of which is the most significant bit, then $a+x$ and $b+x$ differ in these k positions, and at no other position, with probability $1/2^k$. If one of the bit positions where a and b differ is the most significant bit, then this probability is $1/2^{k-1}$.

   See Proposition 2.6.1 below for proof of this assumption[2].

3. Table 2.1 is used to determine the differential probabilities for the approximations of f_{IF} and f_{MAJ}.

Since the XOR and additive differences coincide for the most significant bit, to achieve higher probability, it is advantageous to ensure that many bits in the differential path are MSBs. Based on this observation, we choose $x = 2^{31}$ for SHA-256 and $x = 2^{63}$ for SHA-512 in Table 2.7 and compute the resulting probabilities. An example of illustration of probability calculations is given in Section 2.6.2 next.

Proposition 2.6.1. *If a and b differ in k different bit positions none of which is the most significant bit, then $a+x$ and $b+x$ differ in these k positions, and at no other position, with probability $1/2^k$. If one of the bit positions where a and b differ is the most significant bit, then this probability is $1/2^{k-1}$.*

Proof. In this proof, we denote the i^{th} bit of any binary string y by y_i where the index i is zero for the least significant bit of y.

Note that the required probability is over all random choices of x. Hence we take x to be a uniform random n-bit string and let $x_0, x_1, ..., x_{n-1}$ be uniform and independent bits of x. We can write $x = x_{n-1}x_{n-2}\ldots x_1x_0$ where x_{n-1} is the MSB and x_0 is the LSB.

Let a and b be n-bit strings differing at k positions indexed by i_1, i_2, ..., i_k, and let $i_1 < i_2 < \ldots < i_k$. Let $(a + x) = d$ and let c be the carry-vector for the addition $a+x$. We define the i^{th} bit of c to be carry into the addition of a and x at the i^{th} bit, and define $c_0 = 0$. The addition of a and x is shown in Figure 2.1. Clearly, c_i depends on $x_0, x_1, \ldots, x_{i-1}$ and is independent of x_i. Also, $d_i = a_i \oplus x_i \oplus c_i$.

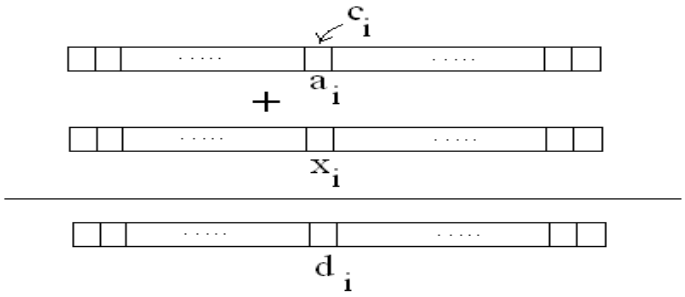
Similarly, define the addition $(b + x) = e$. Also define the event E as follows.

$$E : \begin{cases} d_j \neq e_j & \text{for } j = i_1, i_2, \ldots, i_k; \\ d_j = e_j & \text{otherwise.} \end{cases} \tag{2.6.1}$$

Consider the situation $x_{i_1} = c_{i_1}$. Since a and b are different at this bit position and equal at positions $j < i_1$, the carry vector for the addition $(b + x)$ will be same as that for $(a + x)$ till the position i_1. Now since

[2]We thank an anonymous reviewer of the thesis for the suggestion of providing the proof of this assumption.

Figure 2.1: Addition of bit-strings a and x.



$a_{i_1} \neq b_{i_1}$, $x_{i_1} = c_{i_1}$ and $a_j = b_j$ for $i_1 < j < i_2$, the result of the two additions $(a + x)$ and $(b + x)$ will both have the same carry vectors for all positions j such that $i_1 < j < i_2$. With this insight, we define the events E_j as $E_j : (x_j = c_j)$ for $j = i_1, i_2, \ldots i_k$. Since x_{i_j} is a uniformly distributed bit which is independent of c_{i_j} and of all the previous x_i's; and E_{i_j} is the event $(x_{i_j} = c_{i_j})$, we get that $Pr[E_{i_j}|(E_{i_1} \wedge \ldots \wedge E_{i_{j-1}})] = Pr[E_{i_j}]$. Finally, we note that the required event E can be expressed in terms of these events E_j's as $E = E_{i_1} \wedge E_{i_2} \wedge \ldots \wedge E_{i_k}$.

Hence the required probability can be computed as follows.

$$\begin{aligned}
Pr[E] &= Pr[E_{i_1} \wedge E_{i_2} \wedge \ldots \wedge E_{i_k}] \\
&= Pr[E_{i_1}] \times Pr[E_{i_2}|E_{i_1}] \times \ldots \times Pr[E_{i_k}|(E_{i_1} \wedge E_{i_2} \wedge E_{i_{k-1}})] \\
&= Pr[E_{i_1}] \times Pr[E_{i_2}] \times \ldots \times Pr[E_{i_k}] \\
&\qquad \text{(See the discussion above.)} \\
&= \underbrace{\frac{1}{2} \times \frac{1}{2} \times \ldots \times \frac{1}{2}}_{k \text{ terms}} \\
&= \frac{1}{2^k}.
\end{aligned}$$

In case one of the bits where a and b differ is the MSB then the words $a + x$ and $b + x$ will necessarily differ at the MSB and the carry out of the MSB, if any, does not affect the outcome of the required event. Therefore the required probability will not have a factor of $\frac{1}{2}$ contributed by the carry out of the MSB. Hence, in this case the required probability will be $\frac{1}{2^{k-1}}$. This completes the proof of the proposition. ■

We also verified this assumption computationally by repeatedly choosing random values of a and b and taking all possible 32-bit words x to compute $a + x$ and $b + x$. The computed values of the required probability match exactly with those predicted from the proposition.

2.6.2 Illustration of Probability Calculation

First we calculate the probability for Round 2, Case 1 of Table 2.9. For this round the f_{MAJ} function's inputs are registers a_1, b_1 and c_1. Differential value of the three registers is (1,0,0). From Table 2.1, we know that the probability that f_{MAJ} will behave as its first argument, is $\frac{1}{2}$. The f_{IF} function takes as inputs the registers e_1, f_1 and g_1 in this round. The second and the third inputs to f_{IF} have zero differences while the first input has a 1-bit difference. In this table, f_{IF} is being approximated by the middle argument and therefore the desired output difference from f_{IF} is 0. This is the case (1,0,0) of Table 2.1 for the f_{IF} function and in this case it will not propagate the difference with probability $\frac{1}{2}$.

In computing Δa_2, there are 6 bits $\Sigma_0(a_1) \oplus \Sigma_1(e_1)$ to be cancelled with the input 6-bit message word difference. The probability for this to happen is $\frac{1}{2^6}$. For calculating Δe_2, there are 3 bits in $\Sigma_1(e_1)$ to be cancelled with the input message word difference and 3 bits $\Sigma_0(x)$ to be propagated into Δe_2 from input. The cancellation part's probability has already been taken care of while considering cancellation of difference in register a_2, and the propagation part's probability is $\frac{1}{2^3}$. The combined probability due to approximations in a_2 and e_2 calculations is $\frac{1}{2^6} \times \frac{1}{2^3}$. Thus, the probability for the differential path to hold for this round is $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2^6} \times \frac{1}{2^3} = \frac{1}{2^{11}}$. Probabilities for other rounds can be computed similarly.

Table 2.5 shows the calculation of probabilities for Case 1 of Table 2.9. In this table, p_1^i (resp. p_2^i) is the probability for f_{MAJ} (resp. f_{IF}) to behave as its first (resp. second) argument in round i given that the differential path holds for the $(i-1)^{th}$ round. In the same way, p_3^i is the probability for differences in registers a_i and e_i to follow the differential path given that f_{MAJ} and f_{IF} behave correctly in this round; and the previous rounds follow the differential path. The last column in this table is the product of previous 3 column entries and it is the probability for this round given that all the previous rounds follow the differential path.

Let A_i denote the event that the differential path holds for round i given that it already holds up to round $i-1$. Then $\Pr[A_1]$ is the value of column 'Pr' in row 1 of Table 2.5. Now we consider $\Pr[A_2]$ and as discussed above, probability of A_2 given that A_1 holds is that the value of column 'Pr' in row 2 of Table 2.5. Therefore, $\Pr[A_1 \wedge A_2] = \Pr[A_1] \times \Pr[A_2|A_1]$.

In the same way, the probability for the differential path to hold till the 9 rounds is given by:

$$\begin{aligned}\Pr[\text{Diff Path holds}] &= \Pr[A_1 \wedge A_2 \wedge \ \ldots \ \wedge A_9] \\ &= \Pr[A_1] \times \Pr[A_2|A_1] \times \ \ldots \ \times \Pr[A_9|(A_1 \wedge A_2 \ldots \wedge A_8)].\end{aligned}$$

The value in the probability column for the i^{th} row in Table 2.5 is equal to $\Pr[A_i|(A_1 \wedge A_2 \wedge \ \ldots \ \wedge A_{i-1})]$. Thus, the probability for Case 1 of Table 2.9

Table 2.5: Probability calculations for Case 1 of Table 2.9.

Round i	p_1^i	p_2^i	p_3^i	Pr.
1	1	1	1×1	1
2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2^6} \times \frac{1}{2^3}$	$\frac{1}{2^{11}}$
3	$\frac{1}{2}$	$\frac{1}{2^4}$	$\frac{1}{2^{12}} \times 1$	$\frac{1}{2^{17}}$
4	$\frac{1}{2}$	$\frac{1}{2^3}$	$\frac{1}{2^3} \times 1$	$\frac{1}{2^7}$
5	1	$\frac{1}{2^4}$	1×1	$\frac{1}{2^4}$
6	1	$\frac{1}{2}$	$\frac{1}{2^6} \times 1$	$\frac{1}{2^7}$
7	1	$\frac{1}{2}$	1×1	$\frac{1}{2}$
8	1	$\frac{1}{2}$	1×1	$\frac{1}{2}$
9	1	1	1×1	1
Total probability =				$\frac{1}{2^{48}}$

is $\frac{1}{2^{48}}$. The probabilities for other differential paths have been computed similarly.

2.6.3 Comments on the Local Collisions Presented

From Tables 2.9, A.1, A.2 and A.3, it is interesting to note that all approximations of f_{MAJ} by the same linear function have the same differential path. The weight of the differential path increases with the increase in the number of variables in the linear approximation of f_{MAJ}.

The GH local collision (Case A) has the highest probability. It is, however, not necessarily the best possible local collision. This is due to the fact that it has two conflicting conditions and may result in an impossible differential path. We illustrate this point using the conflicting condition for f_{IF}. A 12-round impossible differential path for the GH local collision is shown in Table 2.8. This is obtained by interleaving two GH local collisions with the second one starting at the fourth round of the first one. In terms of the Chabaud-Joux [5] type disturbance vector, the 12-round differential path is given by the vector 1001. Here, $\Delta e_6 = 0$, $\Delta f_6 = x \oplus \Sigma_0(x)$ and $\Delta g_6 = x$. This shows that whatever be the value of x, there will be one bit position where the differential input to f_{IF} is $(0, 1, 1)$. From Table 2.1 we have Δf_{IF} to be 1 with probability 1, whereas the approximation of f_{IF} by $l = 0$ will have $\Delta l = 0$. This shows that although the differential path is valid for the linearized version with f_{IF} approximated by $l = 0$, it fails for the actual round function.

As mentioned earlier, the issue of impossible differential paths was also observed in [34]. They developed techniques for circumventing such impossible paths in their collision search attacks on reduced round SHA-2. On

Table 2.6: Summary of different properties of the local collisions. Wt(DP) provides the weight of the differential path; Wt(MD) provides the weight of the message difference; Pr. provides the probability of the differential path; and NIC provides the number of conflicting conditions. The cases are from Table 2.3 and 2.4. Case A is the GH local collision, rest are new local collisions.

Case	A	B	C	D	E	1	2	3	4	5	6
Wt(DP)	24	24	24	24	24	28	28	28	28	28	28
Wt(MD)	24	29	29	34	34	35	33	35	35	29	35
Pr.	$\frac{1}{2^{42}}$	$\frac{1}{2^{45}}$	$\frac{1}{2^{45}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{51}}$	$\frac{1}{2^{51}}$	$\frac{1}{2^{49}}$	$\frac{1}{2^{49}}$
NIC	2	1	1	1	1	0	0	0	0	0	0
Case	7	8	9	10	11	12	13	14	15	16	
Wt(DP)	28	28	28	28	28	28	36	36	36	36	
Wt(MD)	35	37	35	31	37	35	37	37	43	41	
Pr.	$\frac{1}{2^{52}}$	$\frac{1}{2^{52}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{48}}$	$\frac{1}{2^{51}}$	$\frac{1}{2^{51}}$	$\frac{1}{2^{54}}$	$\frac{1}{2^{54}}$	$\frac{1}{2^{57}}$	$\frac{1}{2^{57}}$	
NIC	0	0	0	0	0	0	0	0	0	0	

the other hand, if we use a local collision such as Case 1, then there are no conflicting conditions. Consequently, no circumvention techniques will be required in collision search attacks. The probability of this local collision is a little lower than the GH local collision, but this is offset by the absence of conflicting conditions. This issue will be discussed in detail in the next chapter.

2.7 Conclusions

In this chapter, we have made a systematic study of the local collisions for the SHA-2 family of hash functions. Conflicting conditions have been identified in the various approximations of the constituent Boolean functions. In particular, we have shown that the previous local collision by Gilbert and Handschuh [22] has one conflicting condition each in the approximation of f_{IF} and f_{MAJ}. We have presented 16 new local collisions with no conflicting conditions though the probabilities are a little lower than the GH local collision. A summary of various features of the different local collisions are given in Table 2.6.

In this chapter, we have not considered the issue of message expansion. Combining message expansion with the new local collisions to obtain (reduced round) collisions for the SHA-2 family is discussed in the next chapter.

Table 2.7: Differential paths for the cases of Table 2.3. Probability calculations are done taking x to be 2^{31} for SHA-256 and 2^{63} for SHA-512. First column represents the round number.

rd	Registers								Case A		Case B		Case C		Case D		Case E	
i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr
1	x	0	0	0	x	0	0	0	x	1	x	1	x	1	x	1	x	1
2	0	x	0	0	$\Sigma_0(x)$	x	0	0	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$
3	0	0	x	0	0	$\Sigma_0(x)$	x	0	$\Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$\Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$x \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$\Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$
4	0	0	0	x	0	0	$\Sigma_0(x)$	x	0	$\frac{1}{2^5}$	x	$\frac{1}{2^5}$	$\Sigma_0(x)$	$\frac{1}{2^8}$	x	$\frac{1}{2^5}$	$\Sigma_0(x)$	$\frac{1}{2^8}$
5	0	0	0	0	x	0	0	$\Sigma_0(x)$	x	$\frac{1}{2^3}$	$\Sigma_0(x) \oplus x$	$\frac{1}{2^6}$	x	$\frac{1}{2^3}$	$x \oplus \Sigma_0(x)$	$\frac{1}{2^6}$	x	$\frac{1}{2^3}$
6	0	0	0	0	0	x	0	0	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$
7	0	0	0	0	0	0	x	0	0	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$
8	0	0	0	0	0	0	0	x	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$
9	0	0	0	0	0	0	0	0	x	1	x	1	x	1	x	1	x	1
Total Probability										$\frac{1}{2^{42}}$		$\frac{1}{2^{45}}$		$\frac{1}{2^{45}}$		$\frac{1}{2^{48}}$		$\frac{1}{2^{48}}$

Table 2.8: A 12-round impossible differential path for the Gilbert-Handschuh local collision. In Round 6, the registers (e, f, g) have values (0,1,1) for some bit, for any non-zero x. These registers are input to f_{IF} which outputs bit 1 with probability 1 in this case. This shows that it can not be approximated by the 0 function.

Round i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
1	x	x	0	0	0	x	0	0	0
2	$\Sigma_0(x) \oplus \Sigma_1(x)$	0	x	0	0	$\Sigma_0(x)$	x	0	0
3	$\Sigma_0(x) \oplus \Sigma_1(x)$	0	0	x	0	0	$\Sigma_0(x)$	x	0
4	x	x	0	0	x	x	0	$\Sigma_0(x)$	x
5	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	0	x	0	0	$x \oplus \Sigma_0(x)$	x	0	$\Sigma_0(x)$
6	$\Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	0	0	x	0	0	$x \oplus \Sigma_0(x)$	x	0
7	0	0	0	0	x	0	0	$x \oplus \Sigma_0(x)$	x
8	x	0	0	0	0	x	0	0	$x \oplus \Sigma_0(x)$
9	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	0	0	0	0	0	x	0	0
10	0	0	0	0	0	0	0	x	0
11	0	0	0	0	0	0	0	0	x
12	x	0	0	0	0	0	0	0	0

Table 2.9: Differential paths for the cases of Table 2.4. Probability calculations are done taking x to be 2^{31} for SHA-256 and 2^{63} for SHA-512.

Round	Registers								Case 1		Case 2		Case 3		Case 4	
i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr
1	x	0	0	0	x	0	0	0	x	1	x	1	x	1	x	1
2	0	x	0	0	$x \oplus \Sigma_0(x)$	x	0	0	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$
3	0	0	x	0	0	$x \oplus \Sigma_0(x)$	x	0	$x \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$\Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$\Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{20}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{20}}$
4	0	0	0	x	0	0	$x \oplus \Sigma_0(x)$	x	$x \oplus \Sigma_0(x)$	$\frac{1}{2^7}$	x	$\frac{1}{2^4}$	$x \oplus \Sigma_0(x)$	$\frac{1}{2^7}$	x	$\frac{1}{2^4}$
5	0	0	0	0	x	0	0	$x \oplus \Sigma_0(x)$	x	$\frac{1}{2^4}$	$\Sigma_0(x)$	$\frac{1}{2^7}$	x	$\frac{1}{2^4}$	$\Sigma_0(x)$	$\frac{1}{2^7}$
6	0	0	0	0	0	x	0	0	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$
7	0	0	0	0	0	0	x	0	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$
8	0	0	0	0	0	0	0	x	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$
9	0	0	0	0	0	0	0	0	x	1	x	1	x	1	x	1
Total Probability										$\frac{1}{2^{48}}$		$\frac{1}{2^{48}}$		$\frac{1}{2^{51}}$		$\frac{1}{2^{51}}$

Chapter 3

Using Linearized Local Collisions in Attacking Reduced Round SHA-256

3.1 Introduction

In the last chapter we made a systematic study of linearized local collisions for the SHA-2 family. In this chapter, we utilize the local collisions discussed earlier.

We make two contributions in this work. First we construct a 18-round colliding DP using one of the new local collisions from the previous chapter. Then we describe message modification techniques to find messages following this DP. Using these techniques, we provide an algorithm to generate pairs of messages which collide for the actual SHA-256 reduced to 18 rounds. Our second contribution is to present DPs for 19 to 23 rounds of SHA-256. In obtaining these DPs, we use coding theoretic methods in a novel way.

Using linearized local collisions, there were no colliding DPs known for SHA-256 beyond 18-rounds. Previously known best DP was for near collision for 19-round SHA-256 and it used 23 local collisions. In contrast, our 19-round DP uses only 15 local collisions and is an exact collision path. All the 15 local collisions start in the same word and therefore this DP can also be seen as consisting of a single local collision with the starting word difference having a weight of 15 bits. In addition there are no conflicting conditions caused by the f_{IF} and f_{MAJ} functions for the DPs reported here. Therefore the search for actual colliding message pairs following these paths can be much easier.

We also show that neutral bit technique may not be of much help in finding actual colliding pair of messages for SHA-256 while message modification methods seem to hold much more promise.

3.2 Linearized Local Collisions in SHA-256

Linearized local collisions were studied in the previous chapter. We use two of these local collisions in the present work. The first is due to Gilbert and Handschuh [22] and the second is one of the 16 new local collisions from the last chapter. From among the 16, we choose the 5^{th} local collision (Case 5 in Table A.1) because of the following two reasons :

1. It is one of the 4 for which obtaining an 18-round collision is easier than the other 12 (the others being 7^{th}, 14^{th} and 16^{th}). This issue is explained a little later.

2. It has the highest probability among these 4.

We call the two local collisions the GH local collision and the SS_5 local collision respectively. The other three local collisions are denoted by SS_7, SS_{14} and SS_{16}.

As already studied, the following approximations were used in these local collisions:

1. Operator + is approximated by $\oplus$.

2. In GH, f_{IF} and f_{MAJ} are approximated by zero function.

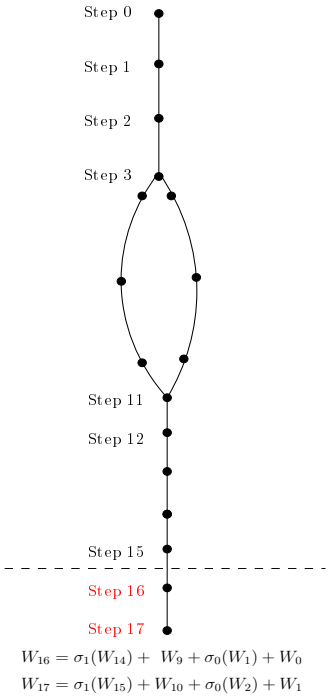
3. In SS_5, f_{IF} and f_{MAJ} are approximated by their middle arguments.

All the local collisions mentioned above are:

- GH : $\{x, \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(\Sigma_1(x)),\ 0,\ x,\ \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$
- SS_5 : $\{x, \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(\Sigma_1(x)),\ \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$
- SS_7 : $\{x, x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x)),\ x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$
- SS_{14} : $\{x, x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ x \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x)),\ \Sigma_1(x),\ \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$
- SS_{16} : $\{x, \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x)),\ x \oplus \Sigma_1(x),\ x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ x \oplus \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$

For differential paths, along with the message word differences, corresponding to these local collisions, refer to Tables A.1, A.2 and A.3. Note that in all the above local collisions, once a starting message difference x is chosen, next 8 words must have the difference in accordance with the chosen local collision.

Figure 3.1: Extending a single local collision to 18 rounds.



3.3 Attacking 18-Round SHA-256

The essential idea of obtaining an 18-round collision is shown in Figure 3.1. The goal is to have a single local collision within the first sixteen free words and then ensure that the first two expanded message words W_{16} and W_{17} do not have any differences. Such an idea has already been used in [34]. We describe this for clarity of exposition.

First of all, note that any local collision under consideration spans 9 rounds and the message expansion of SHA-256 does not play any role in the first 16 rounds. Therefore if a local collision spans from Round i to Round $(i+8)$, and if we take $\Delta W_0 = \Delta W_1 = \ldots = \Delta W_{i-1} = \Delta W_{i+9} = \Delta W_{i+10} = \ldots = \Delta W_{15} = 0$, we get a differential path for 16-round collision for SHA-256.

The issue of message expansion is not considered in obtaining the 16-round colliding differential path described above. Next we tackle two rounds of the message expansion.

Message expansion rule for W_{16} and W_{17} are given by :

$$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0 \quad (3.3.1)$$

$$W_{17} = \sigma_1(W_{15}) + W_{10} + \sigma_0(W_2) + W_1 \quad (3.3.2)$$

Let a local collision $\mathcal{L}$ start at Round 3 and hence end at Round 11. This

Table 3.1: 18-round linear DP for SHA-256. Only 1 SS_5 local collision is used to build this path.

Round i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0-2	0	0	0	0	0	0	0	0	0
3	80000000	80000000	0	0	0	80000000	0	0	0
4	22140240	0	80000000	0	0	20040200	80000000	0	0
5	42851098	0	0	80000000	0	80000000	20040200	80000000	0
6	22140240	0	0	0	80000000	0	80000000	20040200	80000000
7	0	0	0	0	0	80000000	0	80000000	20040200
8	22140240	0	0	0	0	0	80000000	0	80000000
9	0	0	0	0	0	0	0	80000000	0
10	0	0	0	0	0	0	0	0	80000000
11	80000000	0	0	0	0	0	0	0	0
12-17	0	0	0	0	0	0	0	0	0

local collision defines the 9 word differences $\Delta W_3, \Delta W_4, \ldots \Delta W_{11}$. The first round of the local collision corresponds to ΔW_3 and the 9^{th} round corresponds to ΔW_{11}. Taking the differentials of all the message words outside the span of the local collision to be zero, the differential path for $\mathcal{L}$ will have $\Delta W_0 = \Delta W_1 = \Delta W_2 = \Delta W_{12} = \Delta W_{13} = \Delta W_{14} = \Delta W_{15} = 0$.

Note that $\Delta W_i = 0$ means that $W_i = W'_i$. Since $\Delta W_0 = \Delta W_1 = \Delta W_{14} = 0$ for $\mathcal{L}$, from (3.3.1), W_{16} and W'_{16} may be different only due to the differences in W_9 and W'_9.

ΔW_9 corresponds to the 7^{th} round word difference for $\mathcal{L}$. If $\mathcal{L}$ is chosen such that its 7^{th} round word difference is zero, then $W_9 = W'_9$. Therefore even after the message expansion recursion is used, we will have $W_{16} = W'_{16}$. This results in a 17-round colliding differential path for SHA-256.

Similarly, if the 8^{th} message word difference for $\mathcal{L}$ is zero, then by (3.3.2), $W_{17} = W'_{17}$. This results in a 18-round colliding differential path for SHA-256.

Both the 17 and the 18-round DPs discussed above use just one local collision. To increase the probability of this DP for the case of real SHA-256, we can take starting messages differing in only 1 bit.

All the local collisions listed in the previous section have the 7^{th} and the 8^{th} message word differences zero. Therefore any one of them can be used to obtain 18-round colliding differential path for SHA-256 easily. We list one of these differential paths in Table 3.1. This 18-round colliding DP is also a 17-round colliding DP for SHA-256.

3.4 Message Modification Techniques

We have used XOR differences for registers and message words in the differential path for reduced round SHA-256. The differential path in Table 3.1 is obtained by using linearized SHA-256. However our aim is to obtain a pair of messages which follows this differential path for real SHA-256. The probability for this to happen for random messages is 2^{-49} for 18-round SHA-256

(Refer to Case 5 of Table A.1 for the probability estimate). If the message-pair satisfies certain conditions then the probability of the differential path can be increased significantly. We list conditions on the registers and the message words which help in finding messages following the 18-round differential path shown in Table 3.1 when actual SHA-256 is used. These conditions try to ensure that the functions f_{IF} and f_{MAJ} both behave like their middle arguments, and that + behaves like $\oplus$. These conditions are shown in Table 3.2. Sufficient conditions for 9-round SHA-256 collision have also been given in [25], Table 3. We next highlight the advantages of our conditions with those in [25].

1. The conditions in [25] are for only 9-round collision in SHA-256. Our conditions are for 18-round collision in SHA-256.

2. The GH local collision is used in [25] whereas we use the SS_5 local collision. We also provide complete details on the derivation of the conditions required for the SS_5 local collision. It is possible to use the method described in this work to derive conditions for 18-round SHA-256 collisions using any other LC.

3. In [25], the conditions are claimed to be "sufficient" but it is not clear if satisfying them will immediately lead to a collision. The conditions that we identify are not claimed to be sufficient. We only note that satisfying them will increase the probability of finding colliding message pairs.

3.4.1 Explanation of Conditions in Table 3.2

$\Delta W_k = 0$ for rounds k=0, 1 and 2 and hence there are no restrictions due to these rounds. In Round 3, although $\Delta W_3 \neq 0$, the difference is only in the most significant bit. The + and $\oplus$ behave the same with probability 1 for a difference in MSB, so even Round 3 does not impose any restrictions. Hence conditions are needed to tackle the proper differential behavior for the message pair only from Round 4 onwards.

Conditions Due to f_{MAJ} and f_{IF} : In Round 4, f_{MAJ} has inputs a_3, b_3 and c_3 with $\Delta a_3 =$ `0x80000000`. In SS_5 local collision f_{MAJ} is approximated by its middle argument, which will happen if $b_3^{31} = c_3^{31}$. Similarly the f_{IF} function having arguments e_3, f_3 and g_3 will behave like its middle argument if $f_3^{31} = g_3^{31}$.

Table 3.2: Conditions for the 18-round DP in Table 3.1. x^i denotes i^{th} bit of a 32-bit quantity x. $\overline{x}$ denotes the bitwise negation of x which can be a 32-bit or a 1 bit quantity. Operator + is addition modulo 2^{32} and operator * is logical 'AND' of 2 single bits. Both these operators are used in rounds 6 and 8.

Round k	Due to f_{MAJ}	Pr.	Due to f_{IF}	Pr.	Due to a_k	Pr.	Due to e_k	Pr.	Round Pr.
0-3	-	-	-	-	-	-	-	-	1
4	$b_3^{31} = c_3^{31}$	$\frac{1}{2}$	$f_3^{31} = g_3^{31}$	$\frac{1}{2}$	$\overline{W_4^i} = (\Sigma_0(a_3))^i$; $i = 9, 18, 29$ $\overline{W_4^i} = (\Sigma_1(e_3))^i$; $i = 6, 20, 25$	$\frac{1}{2^6}$	bit differences ΔW_4^i; $i = 9, 18, 29$ propagate into e_4	$\frac{1}{2^3}$	$\frac{1}{2^{11}}$
5	$a_4^{31} = \overline{c_4^{31}}$	$\frac{1}{2}$	$e_4^{31} = 1$, $e_3^i = f_3^i$; $i = 9, 18, 29$	$\frac{1}{2^4}$	$\overline{W_5^i} = (\Sigma_1(e_4))^i$; $i = 3, 4, 7, 12, 16, 18, 23, 25, 30$	$\frac{1}{2^9}$	-	-	$\frac{1}{2^{14}}$
6	$a_5^{31} = b_5^{31}$	$\frac{1}{2}$	$e_5^{31} = 1$; $i = 9, 18, 29$ $e_4^{31} = e_5^{\overline{31}} * e_3^{31}$	$\frac{1}{2^4}$	$\overline{W_5^i} = (\Sigma_1(e_5) + f_5)^i$; $i = 6, 9, 18, 20, 25, 29$	$\frac{1}{2^6}$	-	-	$\frac{1}{2^{11}}$
7	-	-	$e_6^i = 1$; $i = 9, 18, 29, 31$	$\frac{1}{2^4}$	-	-	-	-	$\frac{1}{2^4}$
8	-	-	$e_6^{31} = e_7^{\overline{31}} * e_5^{31}$	$\frac{1}{2}$	$\overline{W_8^i} = (\Sigma_1(e_7) + h_7)^i$; $i = 6, 9, 18, 20, 25, 29$	$\frac{1}{2^6}$	-	-	$\frac{1}{2^7}$
9	-	-	$e_8^{31} = 1$	$\frac{1}{2}$	-	-	-	-	$\frac{1}{2}$
10	-	-	$e_9^{31} = 1$	$\frac{1}{2}$	-	-	-	-	$\frac{1}{2}$
11-17	-	-	-	-	-	-	-	-	1
								Prob.	$\frac{1}{2^{49}}$

Conditions Due to Register a_4 : Once the two boolean functions are approximated by their middle arguments, register a_4 is evaluated for both the messages as follows :

$$\begin{aligned} a_4 &= \Sigma_0(a_3) + b_3 + \Sigma_1(e_3) + f_3 + h_3 + K_4 + W_4 \text{ and} \\ a'_4 &= \Sigma_0(a'_3) + b'_3 + \Sigma_1(e'_3) + f'_3 + h'_3 + K_4 + W'_4 \end{aligned}$$

Registers a_3 and a'_3 (resp. e_3 and e'_3) differ in their MSB, and the operator Σ_0 (resp. Σ_1) expands this difference to 3 bit positions 6, 20 and 25 (resp. 9, 18 and 29). The word difference ΔW_4 at this round has been chosen to differ in these 6 bit positions (namely 6, 20, 25, 9, 18 and 29) with the aim of cancelling these differences.

The cancellation will happen as desired if :

1. The difference of words W_4 and W'_4 is opposite to the difference in words $\Sigma_0(a_3)$ and $\Sigma_0(a'_3)$ on bit positions 9, 18 and 29. For example, if $(\Sigma_0(a_3))^i = 1$ and $(\Sigma_0(a'_3))^i = 0$, then we would like $(W_4)^i = 0$ and $(W'_4)^i = 1$ so that $W_4 + \Sigma_0(a_3)$ and $W'_4 + \Sigma_0(a'_3)$ are equal at the i^{th} bit position; $i = 9$, 18 and 29.

2. Similarly, $(W_4)^i$ and $(W'_4)^i$ have difference opposite to the difference in $(\Sigma_1(e_3))^i$ and $(\Sigma_1(e'_3))^i$ at bit positions $i = 6$, 20 and 25.

All the 6 bit differences will be cancelled if the conditions shown in Table 3.2, Round 4, column a_k are met. Note that this is not a necessary way of cancelling the differences, other possibilities exist when the sum of the terms in a_4 and a'_4 may behave as desired. In particular, we do not use bit carries in addition modulo 2^{32} to cancel these type of differences like Wang et. al do for SHA-1 [65]. We use XOR differences only, unlike [65] where modular differences are used.

Conditions due to register e_4 : Having cancelled the 6 bit differences to obtain $\Delta(a_4) = 0$, it can be seen that 3 bits from $\Delta(W_4)$ will certainly propagate into $\Delta(e_4)$ because there is no Σ_0 term in calculating e_4 and e'_4. If the differential path is to be followed, then these 3 differing bits in W_4 and W'_4 should not carry forward to other positions. Carry propagation to other bits will cause problems in adjusting the register differences in next rounds since any single bit difference in a or e register is expanded into 3 bit differences by the operators Σ_0 and Σ_1. We have chosen the word differences in next rounds considering these positions by following the L-characteristics. It is possible to allow some bit carries here but it seems that it will only reduce the probability of the differential path.

To complete the analysis of Round 4, we finally look at the difference $\Delta(e_4)$. The registers e_4 and e'_4 are computed as follows:

$$\begin{aligned} e_4 &= d_3 + \Sigma_1(e_3) + f_{IF}(e_3, f_3, g_3) + h_3 + K_4 + W_4, \\ \text{and} \quad e'_4 &= d'_3 + \Sigma_1(e'_3) + f_{IF}(e'_3, f'_3, g'_3) + h'_3 + K_4 + W'_4. \end{aligned}$$

In these two computations, bits 6, 20 and 25 corresponding to Σ_1 rotations of the differing bit 31 in e_3 have already been taken care of while considering a_4. Bit numbers 9, 18 and 29 are the places where W_4 and W'_4 differ and these differences are required to be propagated to Δe_4. Since $d_3 = d'_3$, $h_3 = h'_3$ and $f_{IF}(e_3, f_3, g_3) = f_{IF}(e'_3, f'_3, g'_3)$;

$$\begin{aligned} \text{if we write} \quad rest &= \Sigma_1(e_3) + f_{IF}(e_3, f_3, g_3) + h_4 + K_4, \\ \text{then} \quad e_4 &= rest + W_4, \\ \text{and} \quad e'_4 &= rest + W'_4. \end{aligned}$$

If the i^{th} bit of $rest$ is 0 and there is no carry into the i^{th} bit while addition with W_4 takes place, then the XOR difference $W_4 \oplus W'_4$ will propagate into $e_4 \oplus e'_4$ as desired. Alternately, if the i^{th} bit of $rest$ is 1 and there is a carry into the i^{th} bit while addition with W_4 takes place, then too the XOR difference $W_4 \oplus W'_4$ will propagate into $e_4 \oplus e'_4$.

Thus, either we would like no carry propagation in e_4 and e'_4 at bits 6, 20 and 25 if $rest$ is 0 at these bit positions or we would like carry propagation in both these registers if $rest$ is 1 at these bits. We do not have a deterministic way to ensure this since we do not have complete freedom to choose the registers and the message words as desired at this stage. However, the probability of the carries to happen as desired can be increased if we set other free bits of W_4 and W'_4 according to the following conditions :

1. if $rest^9$ is 0 then $W_4^7 = W_4^8 = 0$.
2. if $rest^9$ is 1 then $W_4^7 = W_4^8 = 1$.
3. if $rest^{18}$ is 0 then $W_4^{10} = W_4^{11} = \ldots = W_4^{17} = 0$.
4. if $rest^{18}$ is 1 then $W_4^{10} = W_4^{11} = \ldots = W_4^{17} = 1$.
5. if $rest^{29}$ is 0 then $W_4^{26} = W_4^{27} = W_4^{28} = 0$.
6. if $rest^{29}$ is 1 then $W_4^{26} = W_4^{27} = W_4^{28} = 1$.

In setting these conditions, we have used the bits between 6, 9, 20 and 9, 18 and 29 which are not restricted.

Similarly we have set conditions for other rounds so that the messages follow the differential path as desired.

3.4.2 Method to Satisfy Conditions in Table 3.2

First four words in the DP are free and hence we choose them randomly. Thereafter, many conditions in Table 3.2 are easy to fulfill as they depend only on word W_k in round k. Some of the conditions on registers can be tackled by suitably choosing the word W_k at that round which we can choose as desired. However, there may be instances when a previously selected message word causes conflicting condition at a later round. As an example, we may not get the bit carry conditions for register e_4 as described previously. Also we wish to have e_6^{31} following a particular pattern at Round 8 whereas this bit has been set at Round 6 itself. In such contradicting cases, we choose another message word randomly at the round before the round where the condition was breaking down. Then we apply message modification techniques from that round onwards and continue the search process for further rounds. We search incrementally proceeding further only when all the conditions at a round are fulfilled and the differential path is as desired. The differential path in Table 3.1 holds with probability 2^{-49}, but with the procedure described above, we are able to get a much higher probability. In fact, Rounds 0 to 7 become very easy to fulfill with the message modification and we are able to satisfy all the conditions till Round 7 in about a minute on an ordinary PC. The only difficult conditions are those imposed due to a_8. We could find a colliding message pair following exact differential characteristic in time varying from about 40 minutes to a couple of hours on an ordinary PC. Repeatedly running the program we could generate many such pairs.

Tables 3.3 and 3.4 show the message pairs found using the techniques described previously. Similar method can be used for finding 9-round pseudo collisions for SHA-256 as well. Since we can already find message pairs colliding for 18-round SHA-256 with the standard IV, the only utility for such an exercise would be to see how easy it becomes to find these pseudo collisions due to the benefits of relaxing the IV conditions. However, we found that the time required to find a 9-round pseudo collision is only marginally less than the time required to find an 18-round collision. We give an example of such a pseudo collision in Table 3.5.

It seems possible to use neutral bits to increase the efficiency of the search for finding message pairs following the given differential path. We experimented with this idea and found that the gains are not significant. Details about our experiments with neutral bits are given next.

Table 3.3: Colliding message pair for 18-round SHA-256 with standard IV. These two messages follow the differential path given in Table 3.1.

M_1	0-3	ccea5c17	53ad1a2d	141db23c	b6acfaa8
	4-7	5ee7fe4d	53c5b764	2bf20d44	87d63bf6
	8-11	63a07869	f305fdea	26ee271f	b973b91c
	12-15	d0f87828	b724a487	a295fa2a	0a67c97a
M_2	0-3	ccea5c17	53ad1a2d	141db23c	36acfaa8
	4-7	7cf3fc0d	1140a7fc	09e60f04	87d63bf6
	8-11	41b47a29	f305fdea	26ee271f	3973b91c
	12-15	d0f87828	b724a487	a295fa2a	0a67c97a

Table 3.4: Another colliding message pair for 18-round SHA-256 with standard IV. These two messages also follow the differential path given in Table 3.1.

M_1	0-3	ed919421	aa75e4fe	8548d0e0	9c1888f7
	4-7	1da3fc3d	a11f7a02	bb463b64	e9b28365
	8-11	323ecf28	8097e497	4343b78b	dc484e91
	12-15	bf588b4b	8401140a	42499da1	f88a3e2e
M_2	0-3	ed919421	aa75e4fe	8548d0e0	1c1888f7
	4-7	3fb7fe7d	e39a6a9a	99523924	e9b28365
	8-11	102acd68	8097e497	4343b78b	5c484e91
	12-15	bf588b4b	8401140a	42499da1	f88a3e2e

Table 3.5: IV and the messages for the 9-round pseudo collision. These two messages follow the differential path given in Table 3.1 starting from Round 3 till Round 11 (i.e. Round 3 in Table 3.1 is Round 1 for this message pair).

registers	a	b	c	d
IV	1b309331	1d07d226	190c04e9	0b413baf
	e	f	g	h
	7c11cf34	2d035d09	76e27935	0fb234e2

M_1	0-2	5ce03f69	a36bfc0b	f99332c1
	3-5	590db302	7c1cd4df	163c2f4b
	6-8	2077b003	29ab9330	efb8306e
M_2	0-2	dce03f69	817ffe4b	bb162259
	3-5	7b19b142	7c1cd4df	34282d0b
	6-9	2077b003	29ab9330	6fb8306e

3.5 Using Neutral Bits to Search for the Colliding Pairs

Once a linear differential path is obtained, message modification or neutral bit technique are used to find message pairs following that characteristic. Biham and Chen [2] introduced the concept of neutral bits and utilized it to attack SHA-0. They made the following two assumptions regarding the SHA-0 hash function.

1. All the 1-neutral bits are also 2-neutral; and
2. The set of neutral bits is about $1/8^{th}$ the size of the set of 2-neutral bits.

The large number of neutral bits obtained in the case of SHA-0 allowed the authors to generate new candidate message pairs conforming to the differential characteristics without any extra effort.

It is suggested in [34] that using neutral bits will make the probability of finding the colliding message pairs for reduced round SHA-256 close to 1. We had tried to use neutral bit technique on random messages to get a message pair colliding for 18-step SHA-256. But the number of neutral bits was too low for random messages pairs that were generated during the search for the right pair. Consequently, we used message modification technique to find the message pairs colliding for 18-step SHA-256 (as already explained in Section 3.4). We next provide details of the number of 1-neutral bits and

Table 3.6: Count of 1-neutral bits and their distribution for the message pair shown in Table 3.3. Rounds 11 to 15 are not shown here since they do not put any restrictions on the messages and hence have all bits neutral.

Round i	0	1	2	3	4	5	6	7	8	9	10	Total
0	32	–	–	–	–	–	–	–	–	–	–	32
1	32	32	–	–	–	–	–	–	–	–	–	64
2	32	32	32	–	–	–	–	–	–	–	–	96
3	32	32	32	32	–	–	–	–	–	–	–	128
4	0	0	0	5	23	–	–	–	–	–	–	28
5	0	0	0	0	4	23	–	–	–	–	–	27
6	0	0	0	0	0	15	26	–	–	–	–	41
7	0	0	0	0	0	2	23	32	–	–	–	57
8	0	0	0	0	0	0	3	21	26	–	–	50
9	0	0	0	0	0	0	2	12	23	32	–	69
10	0	0	0	0	0	0	1	7	17	30	32	87

their distribution in various words of the message pairs for the two colliding message pairs shown in Tables 3.3 and 3.4.

The distribution of neutral bits for the two message pairs is shown in Tables 3.6 and 3.7. The entry in the j^{th} column of the i^{th} row in the two tables denotes the number of 1-neutral bits for the j^{th} message word at round i. The entry '–' means that the particular message word is not available at that round.

As can be expected, all the bits in the first 4 words in both the tables are 1-neutral. However, as soon as the fifth word is chosen, the number of 1-neutral bits comes down to about 30 and surprisingly almost all the bits of first 4 words do not remain 1-neutral anymore. It is not clear that the assumptions made in [2] for SHA-0 also hold for SHA-256. Consequently, the application of neutral bits alone in SHA-256 may not help us in finding colliding message pairs. A combination of message modification and neutral bit technique may be required to attack more number of rounds in the SHA-2 family.

3.6 Using Coding Theoretic Methods to Find Linear DPs

In [41] and [39] coding theoretic techniques were used to search for differential paths in SHA-1. Extension to SHA-2 was mentioned in [34]. We describe

Table 3.7: Count of 1-neutral bits and their distribution for the message pair shown in Table 3.4. Rounds 11 to 15 are not shown here since they do not put any restrictions on the messages and hence have all bits neutral.

Round i	0	1	2	3	4	5	6	7	8	9	10	Total
0	32	–	–	–	–	–	–	–	–	–	–	32
1	32	32	–	–	–	–	–	–	–	–	–	64
2	32	32	32	–	–	–	–	–	–	–	–	96
3	32	32	32	32	–	–	–	–	–	–	–	128
4	0	0	0	5	26	–	–	–	–	–	–	31
5	0	0	0	0	8	23	–	–	–	–	–	31
6	0	0	0	0	0	13	26	–	–	–	–	39
7	0	0	0	0	0	3	22	32	–	–	–	57
8	0	0	0	0	0	1	6	19	26	–	–	52
9	0	0	0	0	0	1	3	15	25	32	–	76
10	0	0	0	0	0	1	1	9	19	31	32	93

a new way of forming parity check equations and then find low weight codewords for the corresponding generator matrix. Each of these codewords can be used to build a differential characteristic for reduced round SHA-256. This method results in tackling up to 23-round reduced SHA-256.

3.6.1 A New Way of Constructing Parity Check Equations

Tackling message expansion in SHA-2 can be a problem. A non-zero value of ΔW_i for $i \geq 16$ necessitates tackling the recursion for message expansion. So one way to avoid this is to ensure that $\Delta W_i = 0$ for $i \geq 16$. Clearly, this cannot work for full SHA-2. But, for round reduced versions, one can find differential paths using this approach, as we describe below.

The technique described below assumes a local collision $\mathcal{L}$. The description is not for one particular local collision. It holds for any local collision. As already studied in Chapter 2, obtaining a particular local collision requires certain linear approximations of the constituents of the SHA-256 round function. This converts the round function into a linear map based on which we define our linear code. Our method of obtaining linear code is different from that described in [39]. One aspect of this difference is that our method can be used only up to 23 rounds (see below), whereas the method in [39] is independent of the number of rounds.

A message consists of 16 32-bit words for a total of 512 bits. We use

the Chabaud-Joux [5] type disturbance vector approach. Let DV = $\{d_0, d_1, d_2, \ldots, d_{255}\}$ be a 256-bit disturbance vector. If $d_i = 1$ then the two initial messages differ in their i^{th} bit, and further message bits differ as per the local collision.

We do not consider a 512-bit DV for the following reason. A local collision defines the differences of 9 words of messages and only the first 16 words of SHA-256 are unrestricted. Thereafter the message words are calculated using the message expansion recurrence. This implies that a local collision can not be started after first 8 rounds without affecting the message expansion.

Let us now describe the linear code that we require. This is done in two rounds. In the first round, we express ΔW_i ($i \geq 16$) in terms of $d_0, \ldots, d_{255}$. In the second round, we define the parity check equations for the code by setting $\Delta W_i = 0$ for $i \geq 16$. Thus, any DV $(d_0, \ldots, d_{255})$ which satisfies these parity check equations is a codeword. Our task then is to look for a low weight codeword as this gives a differential path with a small number of local collisions.

The first task is to express ΔW_i ($i \geq 16$) in terms of $d_0, \ldots, d_{255}$. We describe how this is done. Let $\mathcal{L}$ be a 9-round local collision for SHA-256. For any local collision $\mathcal{L}$, the first word determines the next eight words. We define $\mathcal{L}(x)$ to denote the nine words of message differentials as per the local collision when the initial word difference is x. In terms of the notation above, we can define the Gilbert-Handschuh local collision [22] as $\mathcal{L}(x) = \{x, \Sigma_0(x) \oplus \Sigma_1(x),\ \Sigma_0(\Sigma_1(x)),\ 0,\ x,\ \Sigma_0(x) \oplus \Sigma_1(x),\ 0,\ 0,\ x\}$. Some more examples of local collisions are given in Section 3.2.

Consider the 32-bit vector $(d_0, 0, \ldots, 0)$, where d_0 is treated as a (binary) variable. Note that, following Chabaud-Joux [5], we treat disturbance vector to denote the place where the local collision is started. In this case, $d_0 = 1$ will imply that a local collision is started at the first bit of the first message word. Using the notation defined above, $\mathcal{L}(d_0, 0, \ldots, 0)$ defines the nine 32-bit words affected by the local collision. At this point, only the first nine words have been defined. The rest seven are taken to be zero. For $i \geq 16$, ΔW_i is now obtained using the message recursion. This expresses all the ΔW_is ($i \geq 16$) as linear function of d_0.

Next consider the 32-bit vector $(0, d_1, 0, \ldots, 0)$; use $\mathcal{L}$ to obtain the next eight words and the message expansion recursion to express ΔW_is ($i \geq 16$) as linear function of d_1. Now, for the 32-bit vector $(d_0, d_1, 0, \ldots, 0)$, we can express ΔW_is ($i \geq 16$) as linear function of d_0 and d_1 by XORing the separate ΔW_is corresponding to d_0 and d_1. Clearly, the procedure can be extended to the entire DV $(d_0, \ldots, d_{255})$. The exact details are given in Table 3.8.

It is clear that such codes can be formed as long as there are less than 256 parity check equations. If we apply this procedure up to N rounds (corresponding to round $N-1$), then we will obtain $32(N-16)$ parity check equations. Thus, the maximum N that we can use with this method is

Table 3.8: Algorithm for generating parity check equations for linearized N-round SHA-256.

external LC(x) : accepts a 32 bit input x and returns 9 words of 32 bits conforming to the local collision chosen.

· Set $\Delta W_{final} := (U_0, U_1, \ldots U_{N-1}) \quad U_i \in \{0,1\}^{32}$
· Set $\delta W_{cur} \quad := (V_0, V_1, \ldots V_8) \qquad V_i \in \{0,1\}^{32}$
· Initialize ΔW_{final} and δW_{cur} to all zeros.

For($i = 0$ to 8){
  For($j = 0$ to 31){
    set $D :=$ (0, 0, …, d_{32i+j}, 0, …, 0);
      /* The j^{th} bit of D is given by d_{32i+j}.
         Each $d_n \in \{0,1\}$ is the component of the disturbance
         vector and $D \in \{0,1\}^{32}$ */
    set $\delta W_{cur} :=$ LC(D);
    For($k = i$ to $i + 8$){
      $\Delta W_{final}[k] = \Delta W_{final}[k] \oplus \delta W_{cur}[k - i]$;
    }
  }
}
/* At this point the ΔW_{final} list contains $W_i \oplus W_i'$ for $0 \leq i < 16$ */

· Obtain ΔW_i for $16 \leq i < N$ using linearized message expansion of SHA-256.
· Equate all 32 bits of ΔW_i for $i \geq 16$ to zero to get $32 * (N - 16)$ parity check equations.

$N = 23$. The minimum value of N is clearly 17. Since we already report 18-round collision, we do not consider $N = 17$ and 18. Instead we report differential paths from 19 to 23 rounds.

Methods presented in [4], [31] and [60] are used to search for low weight codewords from the check-matrices (and the corresponding generator matrices) obtained using the algorithm in Table 3.8. Codewords of least weight found and the linear differential path for that codeword are shown in the next section.

3.7 Results

Low weight disturbance vectors are searched for round reduced SHA-256 by using the probabilistic methods described in [4], [31] and [60]. The minimum weights of codewords found are listed in Table 3.9. For 19-round SHA-256 the weight of the codeword found is 15 for both GH and SS_5 local collision. This means that 15 local collisions are interleaved to obtain the 19-round DP. Interestingly, all the 15 local collisions start at the same word for both GH and SS_5. Thus the case of 19-round DP can be considered as consisting of a single local collision starting at Round 3 where the initial message difference is a word with weight 15 bits. Prior to this work, the best known 19-round differential path was for a near collision consisting of 23 GH local collisions [34]. There was no colliding known differential path for 19 or more rounds using the linearized local collision.

For 20 to 23 rounds, no differential path using a linearized local collision is known so far. We provide the first differential paths for these cases using the linearization technique. For 23-round SHA-256, the size of the corresponding generator matrix is 32×256, i.e. there are only 32 codewords of length 256. It is possible to do exhaustive search on this size, hence we did not use the probabilistic methods for this case. For the 23-round case, the reported codeword weight is actually the minimum possible.

Two differential paths for 19-round SHA-256 are provided in Tables 3.10 and 3.11. These DPs have been obtained by using the SS_5 and the GH local collisions respectively. Differential paths for 20 to 23-round SHA-256 are provided in Tables A.4 to A.11 in Section A.2 in the Appendix. All these differential paths have also been obtained by using the SS_5 and the GH local collisions.

Difficulty of Obtaining Message Pairs Following These DPs. Even after minimizing the number of local collisions to obtain the differential paths, the weight of the difference of message words is quite large. There does not appear to be any direct way to obtain message pairs following these characteristics. We tried to obtain message pairs following the 19-round DP

Table 3.9: Summary of results. Least weight of the codeword found using different local collisions. For the 23-round case, the codeword weight is obtained by exhaustive search. For all other cases, methods described in [4], [31] and [60] are used.

Round i	Size of Check matrix	using GH	using SS_5
18	–	1	1
19	96×256	15	15
20	128×256	33	31
21	160×256	45	45
22	192×256	59	60
23	224×256	79	75

shown in Table 3.10, but even for this case, we could not go beyond initial few rounds. The difficulty stems from the fact that there are a large number of bit changes in the messages. This causes more and more conditions on the later message words. Estimating the probability of success of these DPs is also not easy. Unlike the case for 18-round SHA-256 where only one local collision was being utilized, there are multiple interleaved local collisions in these DPs. This makes the estimation of the probability of these DPs much more difficult.

3.8 Conclusions

In this chapter, we first obtained message pairs colliding for 18-round SHA-256. We developed an algorithm for this purpose which utilizes one of the local collisions developed in the previous chapter. We briefly analyzed the applicability of neutral bit technique to speed-up the search for colliding message pairs. Our conclusion is that the direct application of neutral bit technique does not lead to significantly improved results for reduced round SHA-256. We also obtained new differential paths for 19 to 23-round SHA-256. For obtaining these differential paths, we used coding theoretic techniques in a novel way. For these differential paths too, we used the linearized local collisions studied earlier. In the next chapter we study local collisions obtained using non-linear methods.

Table 3.10: 19-round linear DP for SHA-256. There are 15 SS_5 local collisions. This DP can also be seen as a single SS_5 local collision in which weight of ΔW_3 is 15 bits.

Round i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0-2	0	0	0	0	0	0	0	0	0
3	27f42515	27f42515	0	0	0	27f42515	0	0	0
4	bde9c6f8	0	27f42515	0	0	b1c0627b	27f42515	0	0
5	4180045d	0	0	27f42515	0	27f42515	b1c0627b	27f42515	0
6	bde9c6f8	0	0	0	27f42515	0	27f42515	b1c0627b	27f42515
7	0	0	0	0	0	27f42515	0	27f42515	b1c0627b
8	bde9c6f8	0	0	0	0	0	27f42515	0	27f42515
9	0	0	0	0	0	0	0	27f42515	0
10	0	0	0	0	0	0	0	0	27f42515
11	27f42515	0	0	0	0	0	0	0	0
12-18	0	0	0	0	0	0	0	0	0

Table 3.11: 19-round linear DP for SHA-256. There are 15 GH local collisions. This DP can also be seen as a single GH local collision in which weight of ΔW_3 is 15 bits. Note that this DP differs from the one in Table 3.10 at several places. For example in message words and registers at 7^{th} round.

Round i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0-2	0	0	0	0	0	0	0	0	0
3	27f42515	27f42515	0	0	0	27f42515	0	0	0
4	bde9c6f8	0	27f42515	0	0	b1c0627b	27f42515	0	0
5	4180045d	0	0	27f42515	0	0	b1c0627b	27f42515	0
6	0	0	0	0	27f42515	0	0	b1c0627b	27f42515
7	27f42515	0	0	0	0	27f42515	0	0	b1c0627b
8	bde9c6f8	0	0	0	0	0	27f42515	0	0
9	0	0	0	0	0	0	0	27f42515	0
10	0	0	0	0	0	0	0	0	27f42515
11	27f42515	0	0	0	0	0	0	0	0
12-18	0	0	0	0	0	0	0	0	0

Chapter 4

Non-Linear Attacks on Reduced Round SHA-2

4.1 Introduction

In the previous chapters we studied local collisions for the linearized version of SHA-2 compression function and used them to obtain collisions for 18-round SHA-256. An important contribution in the cryptanalysis of reduced round SHA-2 family came recently due to the work of Nikolić and Biryukov [38]. Whereas the prior works [34,35,48] used local collisions valid for the linearized version of SHA-256 from [22] and [47], the work [38] used a local collision which is valid for the actual SHA-256.

In this chapter, we take a general approach to the analysis of SHA-2 family. The set of all possible 9-round local collisions using additive differentials are analyzed using a general and unified framework. Simplification of the expressions are done in a systematic manner which lead us to the local collisions from [38] and [51] as special cases. We will call these NB and SS local collisions respectively.

The study also clarifies several advantages of the SS local collision over the NB local collision. Deterministic constructions of up to 22-round SHA-2 collisions are described using the SS local collision and up to 21-round SHA-2 collisions are described using the NB local collision. For 23 and 24-round SHA-2, we describe a general strategy and then apply the SS local collision to this strategy. The resulting attacks are faster than those proposed by Indesteege et al. using the NB local collision. We provide colliding message pairs for 22, 23 and 24-round SHA-2.

4.1.1 Cross Dependence Equation (CDE)

In the calculation of new register values, at each round of the SHA-2 hash family, registers b, c and d are merely copies of register a values of previous

rounds. Registers e and a are also related since most of the terms in their computation are common. Thus, we note that e_i can be computed solely from the register a values as shown below.

$$\begin{aligned} e_i &= d_{i-1} + \Sigma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i + W_i \\ &= d_{i-1} + a_i - \Sigma_0(a_{i-1}) - f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) \\ &= a_{i-4} + a_i - \Sigma_0(a_{i-1}) - f_{MAJ}(a_{i-1}, a_{i-2}, a_{i-3}). \end{aligned} \tag{4.1.1}$$

This relationship between these two register values, which we call the Cross Dependence Equation (CDE), implies that if the a register values for five consecutive rounds are known then the e register for the last of these rounds can be determined. This fact means that we can control the value of e_i from a_{i-4}, a register value which was computed 4 rounds earlier. Later, we make extensive use of this relation.

A special case of this equation was utilized in Section 6.1 of [51]. The equation in the form above was used in [49]. A consequence of CDE is that it can be used to provide an alternate description of SHA-2 round function. This was independently observed in [26].

The following result can be used to set registers to specific values.

Proposition 4.1.1. *Suppose that $(a_{i-1}, \ldots, h_{i-1})$ are known and α and β are any two n-bit words. Then it is possible to choose W_i such that either $a_i = \alpha$ or $e_i = \beta$. In general, however, using only W_i, it is not possible to simultaneously set both a_i to α and e_i to β.*

Proof. This is an easy consequence of (1.4.1). Consider the equation for a_i. This is given in terms of $(a_{i-1}, \ldots, h_{i-1})$ and W_i. So, if we set

$$\begin{aligned} W_i &= \alpha - (\Sigma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Sigma_1(e_{i-1}) \\ &\quad + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i), \end{aligned}$$

then clearly $a_i = \alpha$ is attained. Similarly for e_i.

Note, however, that using W_i, we cannot simultaneously set the values of both a_i and e_i. ∎

Even though we cannot use Proposition 4.1.1 to simultaneously set the values of a_i and e_i, there is a way out. This way is given by the CDE. Suppose, the values of $a_{i-3}, \ldots, a_i$ have already become fixed, but, a_{i-4} is still free. Then by choosing a suitable value for a_{i-4} we can attain *any* desired value for e_i. Now, using Proposition 4.1.1, we can use W_{i-4} to set a_{i-4} to the required value. So, in effect, we can use W_{i-4} to set e_i to *any* desired value. This is something nice (from a cryptanalytic point of view) and unexpected and we use this feature extensively.

Table 4.1: Some examples of high frequency values of $\delta = \sigma_1(W) - \sigma_1(W-1)$ for SHA-256.

δ	freq_δ	δ	freq_δ
ffff6000	$2^{29} + 2^{26} + 2^{25}$	0000a000	$2^{29} + 2^{26} + 2^{25}$
ffffa000	$2^{29} + 2^{26}$	00006000	$2^{29} + 2^{26}$
ff006001	2^{16}	ff005fff	2^{16}

4.1.2 Differential Properties of σ_1

For the analysis of 23 and 24-round SHA-2, we will need to consider the differential properties of σ_1 with respect to modular addition. The particular property that we require is discussed in this section.

SHA-256. Consider the distribution of $\delta = \sigma_1(W) - \sigma_1(W-1)$ as W ranges over all 2^{32} values. This distribution is highly skewed and was mentioned in Section 7.1 in [51]. Later, it has been independently observed in [26] that δ takes only 6181 values and there are several values of δ which occur for more than 2^{29} or more values of W.

Let freq_δ be the number of W such that $\delta = \sigma_1(W) - \sigma_1(W-1)$. It is quite easy to prepare a list of $(\delta, \mathsf{freq}_\delta)$ values. For each of the 2^{32} values of W, compute $\delta = \sigma_1(W) - \sigma_1(W-1)$. If this δ has been obtained earlier, then increment the frequency for this δ; else insert $(\delta, \mathsf{freq}_\delta = 1)$ into the list. To do this efficiently, we need a suitable index structure for searching and inserting into the list. A height balanced tree (or AVL tree) is the optimal solution; but, for the current application, a simple (data structure) hash technique is good enough and is the technique we implemented. Some values of $(\delta, \mathsf{freq}_\delta)$ are given in Table 4.1.

Note. Interestingly, we have observed that if freq_δ is greater than 2^{16}, then δ is always even.

SHA-512. In this case, $n = 64$ and it is not possible to exhaustively prepare a list of values for $\delta = \sigma_1(W) - \sigma_1(W-1)$ for all possible 2^{64} values of W. Instead, we created a list using 2^{25} randomly chosen values of W. This provides certain values of δ with certain frequencies. From these frequencies we extrapolate to estimate the actual frequency of each delta among all the 2^{64} choices of W. The extrapolation is done in the following manner. If a particular difference δ occurs κ times in 2^{25} random trials, then we expect it to have a frequency freq_δ of about $\kappa \times 2^{64}/2^{25}$. Some of the observed and the

Table 4.2: Some examples of high frequency values of $\delta = \sigma_1(W) - \sigma_1(W-1)$ for SHA-512. The column $\mathsf{freq}_{\mathrm{O}}$ denotes the observed frequencies among 2^{25} random trials of computing δ. The column freq_{δ} contains the extrapolated values of the frequencies for the complete search space of 2^{64}.

δ	$\mathsf{freq}_{\mathrm{O}}$	freq_{δ}	δ	$\mathsf{freq}_{\mathrm{O}}$	freq_{δ}
200000000008	4795491	$2^{61.5}$	8e000000003a9	22	$2^{43.5}$
ffffdfffffffff8	4793201	$2^{61.5}$	fff26000000000c9	22	$2^{43.5}$
1fffffffff8	4792982	$2^{61.5}$	600000000237	18	$2^{43.5}$

extrapolated frequencies are shown in Table 4.2.

4.2 A General Non-Linear Differential Path

We use a differential technique to find a 9-round local collision. The idea is to use modular differentials which was first used for SHA-2 by Nikolić and Biryukov [38]. Given a word w, we define

$$\left.\begin{array}{rcl} x & = & -\delta\Sigma_0^i(w) - \delta f_{MAJ}^i(w,0,0), \\ y & = & -\delta f_{MAJ}^{i+1}(0,w,0), \\ z & = & -\delta f_{MAJ}^{i+2}(0,0,w). \end{array}\right\} \tag{4.2.1}$$

The general differential path and corresponding message differences are shown in Table 4.3. It can be verified that the differential path holds for the stated message differences. We show the first round of the computation, the other rounds are similar. In the $(i+1)$st round we want $\delta a_{i+1} = 0$ and $\delta e_{i+1} = x$. The given values of x and δW_{i+1} ensure that these two conditions hold. Note that the values of the other registers are fixed by the values of

the registers at the ith round.

$$\begin{aligned}
\delta a_{i+1} &= a'_{i+1} - a_{i+1} \\
&= (\Sigma_0(a'_i) + f_{MAJ}(a'_i, b'_i, c'_i) + \Sigma_1(e'_i) + f_{IF}(e'_i, f'_i, g'_i) + h'_i + K'_{i+1} \\
&\quad + W'_{i+1}) - (\Sigma_0(a_i) + f_{MAJ}(a_i, b_i, c_i) + \Sigma_1(e_i) + f_{IF}(e_i, f_i, g_i) \\
&\quad + h_i + K_{i+1} + W_{i+1}) \\
&= (\Sigma_0(a'_i) - \Sigma_0(a_i)) + (f_{MAJ}(a'_i, b'_i, c'_i) - f_{MAJ}(a_i, b_i, c_i)) \\
&\quad + (\Sigma_1(e'_i) - \Sigma_1(e_i)) + (f_{IF}(e'_i, f'_i, g'_i) - f_{IF}(e_i, f_i, g_i)) \\
&\quad + (W'_{i+1} - W_{i+1}) \\
&= \delta\Sigma_0^i(w) + \delta f_{MAJ}^i(w, 0, 0) + \delta\Sigma_1^i(w) + \delta f_{IF}^i(w, 0, 0) + \delta W_{i+1} \\
&= -x + (\delta\Sigma_1^i(w) + \delta f_{IF}^i(w, 0, 0)) + (x - \delta\Sigma_1^i(w) - \delta f_{IF}^i(w, 0, 0)) \\
&= 0 \\
\delta e_{i+1} &= e'_{i+1} - e_{i+1} \\
&= (\Sigma_1(e'_i) + f_{IF}(e'_i, f'_i, g'_i) + h'_i + K'_{i+1} + W'_{i+1}) \\
&\quad - (\Sigma_1(e_i) + f_{IF}(e_i, f_i, g_i) + h_i + K_{i+1} + W_{i+1}) \\
&= (\Sigma_1(e'_i) - \Sigma_1(e_i)) + (f_{IF}(e'_i, f'_i, g'_i) - f_{IF}(e_i, f_i, g_i)) \\
&\quad + (W'_{i+1} - W_{i+1}) \\
&= \delta\Sigma_1^i(w) + \delta f_{IF}^i(w, 0, 0) + \delta W_{i+1} \\
&= \delta\Sigma_1^i(w) + \delta f_{IF}^i(w, 0, 0) + x - \delta\Sigma_1^i(w) - \delta f_{IF}^i(w, 0, 0) \\
&= x.
\end{aligned}$$

The important thing to note about the differential path shown in Table 4.3 is that it puts no restrictions on the actual message words $W_i, \ldots, W_{i+8}$. Starting at any value for the registers a_i to h_i, and using any given non-zero w, and any $W_i, \ldots, W_{i+8}$, we simply run the compression function step-by-step and define the words x, y, z, the respective δW_is and consequently the respective W'_is. All the steps are deterministic and hence with probability one, we obtain W'_is which collide with W_is. This gives rise to a local collision.

Note. We have defined $\delta X = X' - X$ and so $\delta W_i = w$ means $W'_i = W_i + w$; if we had defined δX to be $X - X'$, then W'_i would have been $W_i - w$. Consequently, without loss of generality one can assume $w > 0$.

Specifying the values of (w, x, y, z) completely specifies message differences as well as the differences in the register values at all the rounds. Two special cases for (w, x, y, z) have been used.

Nikolić-Biryukov (NB) [38]. $(w, x, y, z) = (1, -1, 0, 0)$.

Sanadhya-Sarkar (SS) [51]. $(w, x, y, z) = (1, -1, -1, 0)$.

The NB local collision was the first to be proposed and has been used for finding collisions in both [38] and [26,27]. The SS local collision was proposed

Table 4.3: General 9-round nonlinear local collision for the SHA-2 family.

Differential Path									
Round i	δW_i	δa_i	δb_i	δc_i	δd_i	δe_i	δf_i	δg_i	δh_i
$i-1$	0	0	0	0	0	0	0	0	0
i	w	w	0	0	0	w	0	0	0
$i+1$	δW_{i+1}	0	w	0	0	x	w	0	0
$i+2$	δW_{i+2}	0	0	w	0	y	x	w	0
$i+3$	δW_{i+3}	0	0	0	w	z	y	x	w
$i+4$	δW_{i+4}	0	0	0	0	w	z	y	x
$i+5$	δW_{i+5}	0	0	0	0	0	w	z	y
$i+6$	δW_{i+6}	0	0	0	0	0	0	w	z
$i+7$	δW_{i+7}	0	0	0	0	0	0	0	w
$i+8$	δW_{i+8}	0	0	0	0	0	0	0	0

Message Word Differences

$$
\begin{aligned}
\delta W_i &= w;\\
\delta W_{i+1} &= x - \delta\Sigma_1^i(w) - \delta f_{IF}^i(w,0,0);\\
\delta W_{i+2} &= y - \delta\Sigma_1^{i+1}(x) - \delta f_{IF}^{i+1}(x,w,0);\\
\delta W_{i+3} &= z - \delta\Sigma_1^{i+2}(y) - \delta f_{IF}^{i+2}(y,x,w);\\
\delta W_{i+4} &= -w - \delta\Sigma_1^{i+3}(z) - \delta f_{IF}^{i+3}(z,y,x);\\
\delta W_{i+5} &= -x - \delta\Sigma_1^{i+4}(w) - \delta f_{IF}^{i+4}(w,z,y);\\
\delta W_{i+6} &= -y - \delta f_{IF}^{i+5}(0,w,z);\\
\delta W_{i+7} &= -z - \delta f_{IF}^{i+6}(0,0,w);\\
\delta W_{i+8} &= -w.
\end{aligned}
$$

Table 4.4: Different cases for (w, x, y, z).

(I)	(II)	(III)	(IV)
$(w, -w, 0, 0)$	$(w, -w, 0, -w)$	$(w, -w, -w, 0)$	$(w, -w, -w, -w)$
(V)	(VI)	(VII)	(VIII)
$(w, -2w, 0, 0)$	$(w, -2w, 0, -w)$	$(w, -2w, -w, 0)$	$(w, -2w, -w, -w)$

later and was motivated by the analysis done in [38]. But, it turns out that the SS local collision is actually more attractive than the NB local collision. This is due to the fact that the time complexities of collision attacks using the SS local collision is lower than the time complexities of collision attacks using the NB local collision. To understand why this is so, one needs to go through the detailed combinatorial analysis of the SHA-2 round function carried out in this chapter.

4.2.1 Simplifications

The differential path by itself is not useful for obtaining longer round collisions. To do this, we need to simplify the expressions and obtain conditions. This is done using several rules which are actually sufficient conditions. The rules and their consequences are described below.

Simplifying $\delta\Sigma_0$. There is only one occurrence of Σ_0 in all the expressions and that is in the expression for x. In both SHA-256 and SHA-512, Σ_0 is a linear function which is invariant only on 0 and -1. Note that -1 = `ffffffff` for SHA-256 and -1 = `ffffffffffffffff` for SHA-512. Since $\delta\Sigma_0^i(w) = \Sigma_0(a_i + w) - \Sigma_0(a_i)$ an easy way to satisfy this is to ensure that both a_i and $a_i + w$ are either 0 or -1.

Rule 1: Ensure that $\delta\Sigma_0^i(w) = w$ by putting $w = 1$ and $a_i = -1$.

Simplifying Majority. If two of the inputs are equal, then the output of $f_{MAJ}()$ is equal to this input. Based on this observation, we have the following rule.

Rule 2: Simplify each occurrence of f_{MAJ} by making two of the inputs equal.

This rule has several consequences. The function f_{MAJ} is used only in the definitions of x, y and z. Consider, for example x which, after the application of Rule 1, is equal to

$$x = -w - f_{MAJ}(a_i + w, a_{i-1}, a_{i-2}) + f_{MAJ}(a_i, a_{i-1}, a_{i-2}).$$

There are three ways to apply Rule 2 to this occurrence of f_{MAJ}. These are:

1. Set $a_{i-1} = a_{i-2}$ which implies $x = -w$;
2. set $a_{i-1} = a_i + w$, $a_i = a_{i-2}$ which implies that $x = -2w$;
3. set $a_{i-2} = a_i + w$, $a_i = a_{i-1}$ which also implies that $x = -2w$.

So applying Rule 2 to x implies that either $x = -w$ (in which case $a_{i-1} = a_{i-2}$) or $x = -2w$ (in which case either ($a_{i-1} = a_i + w$ and $a_i = a_{i-2}$) or ($a_{i-2} = a_i + w$ and $a_i = a_{i-1}$).

Similar reasoning applies to the expressions for y and z. Now, if we simultaneously apply Rule 2 to all the three occurrences of f_{MAJ}, then there are eight possible values of (w, x, y, z) which are listed as Cases (I) to (VIII) in Table 4.4. The related sufficient conditions are given in Table 4.5.

These sufficient conditions specify certain values for the registers (a_{i-2}, a_{i-1}, a_i, a_{i+1}, a_{i+2}) and (e_{i+1}, e_{i+2}). Actually, the conditions on the a-register values are independent and the conditions on the e-register values are obtained from these values using the CDE. Using Proposition 4.1.1, it is possible to set the values of $(W_{i-2}, \ldots, W_{i+2})$ to ensure that the $(a_{i-2}, \ldots, a_{i+2})$ obtain the required values. Consequently, we can ensure that any of the cases in Table 4.5 can be made to hold with probability one.

Note. If $w = 1$, then Case (I) of Table 4.4 corresponds to the NB local collision and Case (III) of Table 4.4 corresponds to the SS local collision. As we proceed, we will see that the other cases become unusable.

4.2.2 Simplifying δW_{i+4} to δW_{i+7}

The expression for δW_{i+4} involves $\delta\Sigma_1^{i+3}(z)$ and $\delta f_{IF}^{i+3}(z, y, x)$. Joint simplification of the above two quantities is possible by ensuring that both e_{i+3} and $e_{i+3} + z$ are either 0 or -1.

1. If $z = 0$, then e_{i+3} can be either 0 or -1.
2. If $z = -w$, then we choose $e_{i+3} = 0$ if $w = 1$; and $e_{i+3} = -1$ if $w = -1$.

Similarly, simplification of δW_{i+5} is possible by ensuring that both w and $e_{i+4} + w$ are either 0 or -1.

For δW_{i+6} and δW_{i+7} we respectively ensure that e_{i+5} and e_{i+6} are either 0 or -1. The effect of these simplifications are summarized in Tables 4.6 and 4.7. In particular, the simplifying conditions and the resulting values of the respective δWs are shown. The condition on the values of e-register can be achieved by setting the corresponding message word W (see Proposition 4.1.1). So, any of the conditions in Tables 4.6 and 4.7 can be achieved.

Table 4.5: Result of applying Rules 1 and 2. For this table, we have $w = 1$ and $a_i = -1$.

Case	a_{i-2}	a_{i-1}	a_i	a_{i+1}	a_{i+2}	e_{i+2}	e_{i+1}
I	α	α	-1	α	α	$-\Sigma_0(\alpha) + \alpha$	$1 + a_{i-3}$
II(a)	0	0	-1	0	-1	-1	$1 + a_{i-3}$
II(b)	-1	-1	-1	-1	0	1	$1 + a_{i-3}$
III(a)	-1	-1	-1	0	0	0	$2 + a_{i-3}$
III(b)	0	0	-1	-1	-1	1	a_{i-3}
IV(a)	-1	-1	-1	0	-1	-1	$2 + a_{i-3}$
IV(b)	0	0	-1	-1	0	2	a_{i-3}
V(a)	-1	0	-1	0	0	-1	$2 + a_{i-3}$
V(b)	0	-1	-1	-1	-1	1	$1 + a_{i-3}$
VI(a)	-1	0	-1	0	-1	-2	$2 + a_{i-3}$
VI(b)	0	-1	-1	-1	0	2	$1 + a_{i-3}$
VII(a)	-1	0	-1	-1	-1	0	$1 + a_{i-3}$
VII(b)	0	-1	-1	0	0	1	$2 + a_{i-3}$
VIII(a)	-1	0	-1	-1	0	1	$1 + a_{i-3}$
VIII(b)	0	-1	-1	0	-1	-1	$2 + a_{i-3}$

Table 4.6: Summary of simplifying conditions for δW_{i+4} and δW_{i+5}. These simplifications require Rules 1 and 2 and so, in particular $w = 1$ in all these cases.

δW	Condition(s)	Value of δW
δW_{i+4}	$z = 0$, $e_{i+3} = 0$	$-w - x$
	$z = 0$, $e_{i+3} = -1$	$-w - y$
	$w = 1$, $z = -w$, $e_{i+3} = 0$	$e_{i+1} - e_{i+2} + y$
δW_{i+5}	$w = 1$, $e_{i+4} = -1$	$-w - x - y + e_{i+3} - e_{i+2}$

Table 4.7: Summary of simplifying conditions for δW_{i+6} and δW_{i+7}. These simplifications do not require these Rules 1 and 2 and consequently, w can be any n-bit word.

δW	Condition(s)	Value of δW
δW_{i+6}	$e_{i+5} = 0$	$-y - z$
	$e_{i+5} = -1$	$-y - w$
δW_{i+7}	$e_{i+6} = 0$	$-w - z$
	$e_{i+6} = -1$	$-z$

Table 4.8: Message expansion from W_{16} to W_{23}.

δW_{16}	=	$\delta\sigma_1(\delta W_{14})$	+	δW_9	+	$\delta\sigma_0(\delta W_1)$	+	δW_0
δW_{17}	=	$\delta\sigma_1(\delta W_{15})$	+	δW_{10}	+	$\delta\sigma_0(\delta W_2)$	+	δW_1
δW_{18}	=	$\delta\sigma_1(\delta W_{16})$	+	δW_{11}	+	$\delta\sigma_0(\delta W_3)$	+	δW_2
δW_{19}	=	$\delta\sigma_1(\delta W_{17})$	+	δW_{12}	+	$\delta\sigma_0(\delta W_4)$	+	δW_3
δW_{20}	=	$\delta\sigma_1(\delta W_{18})$	+	δW_{13}	+	$\delta\sigma_0(\delta W_5)$	+	δW_4
δW_{21}	=	$\delta\sigma_1(\delta W_{19})$	+	δW_{14}	+	$\delta\sigma_0(\delta W_6)$	+	δW_5
δW_{22}	=	$\delta\sigma_1(\delta W_{20})$	+	δW_{15}	+	$\delta\sigma_0(\delta W_7)$	+	δW_6
δW_{23}	=	$\delta\sigma_1(\delta W_{21})$	+	δW_{16}	+	$\delta\sigma_0(\delta W_8)$	+	δW_7

4.3 Obtaining Up To 22-Round Collisions

The basic idea is the following. Choose a suitable value for i and place the local collision from Rounds i to $i+8$. By placing we mean the following. Ensure that $\delta W_0, \ldots, \delta W_{i-1}$ are all zeros and introduce the required differences in $\delta W_i, \ldots, \delta W_{i+8}$. This creates a collision from Rounds i to $i+8$. Ensure that there are no further disturbances by setting δW_{i+9} to δW_{15} to be zero. This works well if we are interested in up to 16-round collisions.

For obtaining collisions on $r > 16$ rounds, we need to consider the message expansion. The initial words $W_0, \ldots, W_{15}$ are free and from W_{16} onwards, the words are computed using the message expansion recursion given by (1.4.2). For clarity, some word differences are shown in Table 4.8. The differences in the message words introduced in Rounds i to $i+8$ can possibly affect $\delta W_{16}, \delta W_{17}, \ldots, \delta W_{r-1}$. We ensure that the effects of these induced differences can be cancelled out and we have $\delta W_{16} = \delta W_{17} = \cdots = \delta W_{r-1} = 0$. This results in an r-round collision.

18-Round Collisions. Deterministic 18-round collisions are easy to obtain by setting $i = 3$ (i.e., the local collision spans from $i = 3$ to $i + 8 = 11$). So, we necessarily have $\delta W_j = 0$ for $j = 0, 1, 2, 12, 13, 14, 15$.

Additionally, we need to ensure that $\delta W_{16} = \delta W_{17} = 0$. From Table 4.8, we see that in the expression for δW_{16} the only possible non-zero term is $\delta W_9 = \delta W_{i+6}$. Similarly, in the expression for δW_{16}, the only possible non-zero term is $\delta W_{10} = \delta W_{i+7}$. By ensuring that $\delta W_{i+6} = \delta W_{i+7} = 0$, we will obtain $\delta W_{16} = \delta W_{17} = 0$. But, ensuring $\delta W_{i+6} = \delta W_{i+7} = 0$ can be easily done by setting a suitable condition from Table 4.7. For example, if $y = z = 0$, then the setting $e_{i+5} = 0$ and $e_{i+6} = -1$ ensures $\delta W_{i+6} = \delta W_{i+7} = 0$ for any choice of w. Using Proposition 4.1.1, the required values of e_{i+5} and e_{i+6} can be achieved by setting W_{i+6} and W_{i+7} to appropriate values. As a net result, we obtain deterministic 18-round collisions for any value of w.

Table 4.9: Summary of sufficient conditions which ensure deterministic 18 round collisions for all hash function of SHA-2 family. For this table a single local collision is spanned from round $i = 3$ to round $i+8 = 11$. The initial perturbation difference w could be any value.

No.	Condition	Method to satisfy
1	$a_{i+1} = a_{i-1}$	Choose W_{i+1} so that a_{i+1} attains the value a_{i-1}.
2	$a_{i+2} = a_{i+1}$	Choose W_{i+2} so that a_{i+2} attains the value a_{i+1}.
3	$e_{i+5} = 0$	Choose W_{i+5} so that e_{i+5} attains the value 0.
4	$e_{i+6} = -1$	Choose W_{i+6} so that e_{i+6} attains the value -1.

Summary of conditions which will ensure deterministic 18-round collisions for all function of the SHA-2 family is given in Table 4.9. This attack works with a single local collision spanning from Round 3 to Round 11. The local collision is allowed to have any arbitrarily chosen initial perturbation difference w. Example of 18-round SHA-256 and SHA-512 collisions are shown in the Appendix. For more examples with non-zero y and z values, refer to [51].

20-Round Collisions. Set $i = 5$, i.e., the local collision spans from $i = 5$ to $i+8 = 13$, so that $\delta W_j = 0$ for $j = 0, \ldots, 4, 14, 15$. We need to ensure that $\delta W_{16} = \cdots = \delta W_{19} = 0$. From Table 4.8, we see that this can be achieved by setting $\delta W_9 = \delta W_{10} = \delta W_{11} = \delta W_{12} = 0$.

Since $i = 5$, this means that we have to set $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = \delta W_{i+7} = 0$. The conditions for individually setting any of these to 0 are given in Tables 4.6 and 4.7. In the present case, we need to consider how to simultaneously set all of these to 0. In this situation, some conditions become infeasible. More precisely, certain conditions for obtaining $\delta W_{i+4} = 0$ are incompatible with certain conditions for obtaining $\delta W_{i+5} = 0$. The possible conditions for ensuring these two δWs to be zero are given in Table 4.10. In particular, we see that $z = 0$ in all cases. The conditions for setting $\delta W_{i+6} = 0$ and $\delta W_{i+7} = 0$ do not cause any conflict with other conditions. The set of conditions required for setting $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = \delta W_{i+7} = 0$ are summarized in Table 4.11. Again achieving the appropriate values of the a and e-registers can be done using Proposition 4.1.1.

Note. Tables 4.10 and 4.11 show that it is possible to deterministically set all the four δWs to zero in Case (A) which is the NB local collision. Consequently, it is possible to obtain *deterministic* 20-round collision using this local collision. This was not done in [38] but was later mentioned in [51].

Table 4.10: Conditions for setting $\delta W_{i+4} = \delta W_{i+5} = 0$.

Case	w	x	y	z	e_{i+2}	e_{i+3}	e_{i+4}	Extra Condition
A	1	-1	0	0	0	0	-1	Case I
B	1	-1	-1	0	1	0	-1	Case III (b)
C	1	-2	-1	0	1	0	-1	Case VII (b)
D	1	-1	-1	0	0	-1	-1	Case III (a)
E	1	-2	0	0	1	-1	-1	Case V (b)
F	1	-2	-1	0	1	-1	-1	Case VII (b)

Table 4.11: Conditions for setting $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = \delta W_{i+7} = 0$.

A row of Table 4.10
AND
($e_{i+5} = 0$ and $y = -z$) or ($e_{i+5} = -1$ and $y = -w$)
AND
$e_{i+6} = -1$.

21-Round Collisions. Set $i = 6$, i.e., the local collision spans from $i = 6$ to $i + 8 = 14$. We need to ensure that $\delta W_{16} = \cdots = \delta W_{20} = 0$. As in the case of 20-round collision, we set $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = \delta W_{i+7} = 0$ by a suitable set of conditions given by Table 4.11. So, we have $\delta W_j = 0$ for $j = 0, \ldots, 5, 10, 11, 12, 13, 15$. From Table 4.8, we see that if we can now achieve $\delta W_{16} = 0$, then we will have achieved the condition $\delta W_{16} = \cdots = \delta W_{20} = 0$.

From the structure of the differential path shown in Table 4.3, $\delta W_{14} = \delta W_{i+8} = -w$ and so

$$\delta W_{16} \quad = \quad \delta\sigma_1(\delta W_{14}) + \delta W_9. \tag{4.3.1}$$

Consider $\delta W_9 = \delta W_{i+3}$ which by the differential path in Table 4.3 is equal to $z - \delta\Sigma_1^{i+2}(y) - \delta f_{IF}^{i+2}(y, x, w)$. To simplify this, we choose rows from Table 4.10 such that both e_{i+2} and $e_{i+2} + y$ are either 0 or -1. These are rows A and D.

In the case of row D, we have $\delta W_9 = -e_7 + e_6 + 2$; whereas for row A, we get $\delta W_9 = -1$. It is possible to deterministically satisfy the case for row D. However, row A (which is the NB local collision) cannot be used in the attack. This is due to the fact that there does not exist any word X such that $\sigma_0(X) - \sigma_0(X - 1) = -1$ either for SHA-256 or for SHA-512.

Since $i = 6$, the row of Table 4.5 corresponding to row D of Table 4.10 ensure that a_4, a_5, a_6, a_7, a_8 and e_8 are all fixed to particular values. Using

CDE, we can now use a_3 to set e_7 to any specific value and then use a_2 to set e_6 to any specific value.

Now, the following strategy is used to ensure that $\delta W_{16} = 0$. Choose an arbitrary value for W_{14} and compute δ to be

$$\delta = \delta\sigma_1(\delta W_{14}) = \sigma_1(W_{14} + \delta W_{14}) - \sigma_1(W_{14}) = \sigma_1(W_{14} - w) - \sigma_1(W14).$$

Choose W_2 and W_3 to set a_2 and a_3 such that $e_7 - e_6 - 2 = -\delta$. From (4.3.1), it now follows that $\delta W_{16} = 0$. This gives a deterministic 21-round collision.

It is also possible to obtain deterministic 21-round collision by placing the local collision from Rounds 7 to 15. Set $i = 7$ so that the local collision spans rounds $i = 7$ to $i + 8 = 15$. In this case, set $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = 0$ the sufficient condition for this being any row of Table 4.10 AND (($e_{i+5} = 0$, $y = -z$) or ($e_{i+5} = -1$, $y = -w$)). This ensures $\delta W_{11} = \delta W_{12} = \delta W_{13} = 0$. Now

$$\begin{aligned} \delta W_{16} &= \sigma_1(\delta W_{14}) + \delta W_9, \\ \delta W_{17} &= \sigma_1(\delta W_{15}) + \delta W_{10}. \end{aligned}$$

We have $\delta W_{15} = -w$ and by setting $e_{i+6} = 0$, we also have $\delta W_{14} = -w$. Also,

$$\begin{aligned} \delta W_9 = \delta W_{i+2} &= y - \delta\Sigma_1^{i+1}(x) - \delta f_{IF}^{i+1}(x, w, 0) \\ \delta W_{10} = \delta W_{i+3} &= -\delta\Sigma_1^{i+2}(y) - \delta f_{IF}^{i+2}(y, x, w). \end{aligned}$$

To simplify $\delta W_{10} = \delta W_{i+3}$ we choose rows from Table 4.10 such that both e_{i+2} and $e_{i+2} + y$ are either 0 or -1. These are rows A and D (which correspond to the NB and SS local collisions respectively). Similarly, to simplify $\delta W_9 = \delta W_{i+2}$, in row A we choose $e_{i+1} = 0$; and in row D we choose $e_{i+1} = -1$.

The overall strategy is now the following. Choose arbitrary values for W_{14} and W_{15} and compute $\delta_1 = \delta\sigma_1(\delta W_{14})$ and $\delta_2 = \delta\sigma_1(\delta W_{15})$, where $\delta W_{14} = \delta W_{15} = -w$. Now set $\delta W_9 = -\delta_1$ and $W_{10} = -\delta_2$ using W_3 and W_4 to set a_3 and a_4 and hence, using CDE to set e_7 and e_8 to desired values. This can be done deterministically.

We have sketched two ways of achieving deterministic 21-round collisions. In one case, the local collision spans from Round 6 to Round 14 and in the second case, the local collision spans from Round 7 to Round 15. For the first case, only the SS local collision can be used, while in the second case, both the SS and the NB local collisions can be used. The fact that the NB local collision can be used to obtain deterministic 21-round collisions was not mentioned in [38]; it was mentioned in [49].

The sketches above can be developed into detailed algorithms. We do not describe these algorithms. This is because below we describe in details a similar algorithm for constructing deterministic 22-round collisions.

4.3.1 22-Round Collisions

Set $i = 7$ so that the local collision spans from $i = 7$ to $i + 8 = 15$. Use sufficient conditions from Table 4.11 to ensure that $\delta W_{i+4} = \delta W_{i+5} = \delta W_{i+6} = \delta W_{i+7} = 0$. So $\delta W_j = 0$ for $j = 0, \ldots, 6, 11, 12, 13, 14$. If we can now ensure that $\delta W_{16} = \delta W_{17} = 0$, then from Table 4.8, we will have $\delta W_j = 0$ for $j = 18, 19, 20, 21$ which will give rise to a 22-round collision. Under the conditions, we have

$$\left.\begin{array}{rcl} \delta W_{16} & = & \delta W_9 \\ \delta W_{17} & = & \delta\sigma_1(\delta W_{15}) + \delta W_{10}. \end{array}\right\} \tag{4.3.2}$$

So, if we can achieve $\delta W_9 = \delta W_{i+2} = 0$ and $\delta\sigma_1(\delta W_{15}) + \delta W_{10} = 0$, then we are done. Note that $\delta W_{15} = -w = -1$.

First, consider the condition on δW_{17}. To simplify $\delta W_{10} = \delta W_{i+3}$ we need to choose both e_{i+2} and $e_{i+2} + y$ to be 0 or -1. These imply that we have to use either row A or row D of Table 4.10 (which respectively correspond to the NB and the SS local collisions).

In case of row D, we have $\delta W_{10} = -e_8 + e_7 = 2$, whereas in the case of row A, we have $\delta W_{10} = -1$. The computation for row D is as follows. (A similar computation shows the value of δW_{10} for row A.)

$$\begin{array}{rcl} \delta W_{10} & = & -\delta f^9_{IF}(-1,-1,1) - \delta\Sigma_1(-1) \\ & = & -f_{IF}(e_9-1, f_9-1, g_9+1) + f_{IF}(e_9, f_9, g_9) \\ & & -\Sigma_1(e_9-1) + \Sigma_1(e_9) \\ & = & -f_{IF}(e_9-1, e_8-1, e_7+1) + f_{IF}(e_9, e_8, e_7) \\ & & -\Sigma_1(e_9-1) + \Sigma_1(e_9) \\ & = & -f_{IF}(-1, e_8-1, e_7+1) + f_{IF}(0, e_8, e_7) - \Sigma_1(-1) + \Sigma_1(0) \\ & = & -(e_8-1) + e_7 - (-1) + 0 \\ & = & -e_8 + e_7 + 2. \end{array}$$

If we want to use row A, then from (4.3.2) we need to have a value for W_{15} such that $\sigma_1(W_{15}+1) - \sigma_1(W_{15}) = 1$. There is no such value for W_{15} for both SHA-256 and SHA-512. Hence, row A, which correspond to the NB local collision, cannot be used.

So, we use row D which correspond to Case III(a) of Table 4.5. In this case, we see that $e_8 = e_{i+1} = 2 + a_{i-3} = 2 + a_4$. We set $a_4 = -2$, so that $e_8 = 0$ and $\delta W_{10} = e_7 + 2$. Setting a_4 to -2 is done using W_4 as in Proposition 4.1.1.

Choose an arbitrary value for W_{15} and set $\delta = -\delta\sigma_1(\delta W_{15})$ where $\delta W_{15} = -1$. Then use W_3 to set a_3 such that due to CDE, e_7 gets set to a particular value required to ensure that $e_7 + 2 = \delta W_{10} = \delta$, i.e., $e_7 = \delta - 2$. This

computation is as follows.

$$\begin{aligned}\delta = e_7 + 2 &= a_3 + a_7 - \Sigma_0(a_6) - f_{MAJ}(a_6, a_5, a_4) + 2 \\ &= a_3 + (-1) - \Sigma_0(-1) - f_{MAJ}(-1, -1, -2) + 2 \\ &= a_3 - 1 - (-1) - (-1) + 2 \\ &= a_3 + 3.\end{aligned}$$

So, using W_3 we need to set $a_3 = \delta - 3$.

Now consider the condition on δW_{16} given in (4.3.2), i.e., the condition $\delta W_9 = 0$. Up to this point, the values of a_3 to a_6 have been fixed as follows: $a_3 = \delta - 3$, $a_4 = -2$, $a_5 = -1$, $a_6 = -1$. Noting that $i = 7$ and row D correspond to $(w, x, y, z) = (1, -1, -1, 0)$, from Table 4.3, we have

$$\begin{aligned}\delta W_9 = \delta W_{i+2} &= y - \delta\Sigma_1^{i+1}(x) - \delta f_{IF}^{i+1}(x, w, 0) \\ &= -y - \Sigma_1(e_{i+1} + x) + \Sigma_1(e_{i+1}) \\ &\quad -f_{IF}(e_{i+1} + x, e_i + w, e_{i-1}) + f_{IF}(e_{i+1}, e_i, e_{i-1}) \\ &= -1 - (-1) + 0 - f_{IF}(-1, e_7 + 1, e_6) + f_{IF}(0, e_7, e_6) \\ &= -e_7 - 1 + e_6 \\ &= 2 - \delta - 1 + e_6.\end{aligned}$$

To obtain $\delta W_9 = 0$, we need to have $e_6 = \delta - 1$. Using CDE, we have

$$\begin{aligned}e_6 &= a_6 + a_2 - \Sigma_0(a_5) - f_{MAJ}(a5, a_4, a_3) \\ &= -1 + a_2 - (-1) - f_{MAJ}(-1, -2, \delta - 3)\end{aligned}$$

So, setting $a_2 = \delta - 1 + f_{MAJ}(-1, -2, \delta - 3)$ ensures, $e_6 = \delta - 1$ as required. This completes the description. All of the above steps can be written more explicitly in an algorithmic form. We provide this below.

Algorithm to Obtain 22-Round Collisions. We define two functions which return the required message word W_i to set the register value a_i or e_i to desired values, say `desired_a` and `desired_e`, at Round i. (See Proposition 4.1.1.) Equation 1.4.1 provides the definitions of these two functions.

1. `W_to_set_register_A(Round` i`, desired_a, Current State` $\{a_{i-1}, b_{i-1}, \ldots, h_{i-1}\}$`) : = (desired_a` $-\Sigma_0(a_{i-1}) - f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) -\Sigma_1(e_{i-1}) - f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) - h_{i-1} - K_i$`)`

2. `W_to_set_register_E(Round` i`, desired_e, Current State` $\{a_{i-1}, b_{i-1}, \ldots, h_{i-1}\}$`) : = (desired_e` $-d_{i-1} - \Sigma_1(e_{i-1}) - f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) -h_{i-1} - K_i$`)`

Using these functions, the complete algorithm to obtain message pairs leading to deterministic 22-round collisions for SHA-2 family is described in Table 4.12.

Table 4.12: Deterministic algorithm to obtain message pairs leading to collisions for 22-round SHA-2.

First Message words:

1. Select W_0, W_1, W_{14} and W_{15} randomly.
2. Set $\texttt{DELTA} = \sigma_1(W_{15}) - \sigma_1(W_{15} - 1)$.
3. Run Rounds 0 and 1 of hash evaluation to define $\{a_1, b_1, \ldots h_1\}$.
4. Choose $W_2 =$ `W_to_set_register_A(2,` $\texttt{DELTA} - 1 + f_{MAJ}(-1, -2, \texttt{DELTA} - 3)$, $\{a_1, b_1, \ldots h_1\}$).
5. Run Round 2 of hash evaluation to define $\{a_2, b_2, \ldots h_2\}$.
6. Choose $W_3 =$ `W_to_set_register_A(3,` $\texttt{DELTA} - 3$, $\{a_2, b_2, \ldots h_2\}$).
7. Run Round 3 of hash evaluation to define $\{a_3, b_3, \ldots h_3\}$.
8. Choose $W_4 =$ `W_to_set_register_A(4,` -2, $\{a_3, b_3, \ldots h_3\}$).
9. Run Round 4 of hash evaluation to define $\{a_4, b_4, \ldots h_4\}$.
10. Choose $W_5 =$ `W_to_set_register_A(5,` -1, $\{a_4, b_4, \ldots h_4\}$).
11. Run Round 5 of hash evaluation to define $\{a_5, b_5, \ldots h_5\}$.
12. Choose $W_6 =$ `W_to_set_register_A(6,` -1, $\{a_5, b_5, \ldots h_5\}$).
13. Run Round 6 of hash evaluation to define $\{a_6, b_6, \ldots h_6\}$.
14. Choose $W_7 =$ `W_to_set_register_A(7,` -1, $\{a_6, b_6, \ldots h_6\}$).
15. Run Round 7 of hash evaluation to define $\{a_7, b_7, \ldots h_7\}$.
16. Choose $W_8 =$ `W_to_set_register_A(8,` 0, $\{a_7, b_7, \ldots h_7\}$).
17. Run Round 8 of hash evaluation to define $\{a_8, b_8, \ldots h_8\}$.
18. Choose $W_9 =$ `W_to_set_register_A(9,` 0, $\{a_8, b_8, \ldots h_8\}$).
19. Run Round 9 of hash evaluation to define $\{a_9, b_9, \ldots h_9\}$.
20. Choose $W_{10} =$ `W_to_set_register_E(10,` -1, $\{a_9, b_9, \ldots h_9\}$).
21. Run Round 10 of hash evaluation to define $\{a_{10}, b_{10}, \ldots h_{10}\}$.
22. Choose $W_{11} =$ `W_to_set_register_E(11,` -1, $\{a_{10}, b_{10}, \ldots h_{10}\}$).
23. Run Round 11 of hash evaluation to define $\{a_{11}, b_{11}, \ldots h_{11}\}$.
24. Choose $W_{12} =$ `W_to_set_register_E(12,` -1, $\{a_{11}, b_{11}, \ldots h_{11}\}$).
25. Run Round 12 of hash evaluation to define $\{a_{12}, b_{12}, \ldots h_{12}\}$.
26. Choose $W_{13} =$ `W_to_set_register_E(13,` -1, $\{a_{12}, b_{12}, \ldots h_{12}\}$).

Second message words:

27. Define $\delta W_i = 0$ for $i \in \{0, 1, 2, 3, 4, 5, 6, 9, 11, 12, 13, 14\}$.
28. Define $\delta W_7 = 1$ and $\delta W_{15} = -1$.
29. Define $\delta W_8 = -1 - f_{IF}(e_7 + 1, f_7, g_7) + f_{IF}(e_7, f_7, g_7) - \Sigma_1(e_7 + 1) + \Sigma_1(e_7)$.
30. Define $\delta W_{10} = -f_{IF}(e_9 - 1, f_9 - 1, g_9 + 1) + f_{IF}(e_9, f_9, g_9) - \Sigma_1(e_9 - 1) + \Sigma_1(e_9)$.
31. Compute $W'_i = W_i + \delta W_i$ for $0 \leq i \leq 15$.

A Remark on the NB Local Collision. We have mentioned that if we place the local collision from Rounds 7 to 15, then row A of Table 4.10 cannot be used to obtain a deterministic 22-round collision. Row A corresponds to the NB local collision.

We considered the issue of whether it is possible to place the NB local collision from Rounds 8 to 16 to obtain a 22-round collision (which may not be deterministic). In this case, the local collision will end at Round 16 and hence $\delta W_{16} = -1$. Recall from Table 4.8, that a difference in δW_{16} will affect δW_{18}. We would like to have $\delta W_{18} = 0$ so as to ensure that there are no differences after the local collision ends. Again from Table 4.8 and the fact that the local collision spans Rounds 8 to 16, to achieve $\delta W_{18} = 0$, we need to have $\delta\sigma_1(\delta W_{16}) + \delta W_{11} = 0$.

More generally, we considered the situation, where the NB local collision spans Rounds i to $(i + 8)$, with $i \geq 8$ and we require $\delta W_{i+10} = 0$. From Table 4.8, the last condition is achieved if $\delta\sigma_1(\delta W_{i+8}) + \delta W_{i+3} = 0$. Note that $\delta W_{i+8} = -1$.

For SHA-512, using the NB local collision makes achieving the condition $\delta\sigma_1(\delta W_{i+8}) + \delta W_{i+3} = 0$ difficult. This is because of the fact that there is a "gap" in the values of $|\delta W_{i+3}|$ and $|\delta\sigma_1(\delta W_{i+8})|$. In Section 4.6, we show that the probability of $|\delta W_{i+3}| \geq 2^j$ is less than $1/2^{j-1}$; and for any 64-bit value for W_{i+8}, $|\sigma_1(W_{i+8}) - \sigma_1(W_{i+8} - 1)| \geq 2^{42} + 2^{39} + 2^{38} + 2^{36} - 2^3$. As a consequence, to achieve $\delta\sigma_1(\delta W_{i+8}) + \delta W_{i+3} = 0$, we need to have $|\delta W_{i+3}| > 2^{42}$, an event which occurs with probability less than 2^{-41}.

The above probability computation is over uniform random choices of W_{i+8} and W_{i+3}. In fact, this was one of the factors that had led us to focus only on the SS local collision. It was shown in [26] that the NB local collision can be used to obtain 23 and 24-round SHA-512 collision. However, the time complexities of the NB local collision attack is more than that of the SS local collision attack. This fact is possibly attributable to the "gap" in the values of $|\delta W_{i+3}|$ and $|\delta\sigma_1(\delta W_{i+8})|$ mentioned above.

4.4 A General Idea for Obtaining 23 and 24-Round Collisions

Obtaining deterministic collisions up to 22 rounds did not require the (single) local collision to extend beyond Round 15. For obtaining collisions for more number of rounds, we will need to start the local collision at Round 8 (or further) and hence the local collision will end at Round 16 (or further). This will require us to analyze the message expansion more carefully.

For obtaining collisions up to 22 rounds, we also needed to consider message expansion. But, we ensured that there were no differences in message words from Round 16 onwards. However, now that we consider the local

collision to end at Round 16 (or further), this will necessarily mean that one or more δW_i (for $i \geq 16$) will be non-zero. This will require a modification of the strategy followed so far. Instead of requiring $\delta W_i = 0$ for $i \geq 16$, we will require $\delta W_i = 0$ for a few i's after the local collision ends. So, supposing that the local collision ends at Round 16 and we want a 23-round collision, then δW_{16} is necessarily $-w$ and we will require $\delta W_{17} = \cdots = \delta W_{22} = 0$.

4.4.1 A Class of Local Collisions

A local collision of the type shown in Table 4.3 is completely determined by the values of w, x, y and z which in turn determine the values of δW_i to δW_{i+8}. We need to consider some special values for the δWs. Let

$$(\delta W_i, \ldots, \delta W_{i+8}) \;=\; (w, -w, \delta_1, \delta_2, 0, 0, 0, u, -w) \text{ with } w = 1. \quad (4.4.1)$$

The value of u is either 0 or w and the values of δ_1 and δ_2 will be explained later. Using the form of the δWs from Table 4.3, Equation 4.4.1 gives rise to the following 9 equations. We will refer to them as (4.4.2.1) to (4.4.2.9).

$$\left.\begin{array}{lll}
(1) & \delta W_i = & = w; \\
(2) & \delta W_{i+1} = x - \delta\Sigma_1^i(w) - \delta f_{IF}^i(w,0,0) & = -w; \\
(3) & \delta W_{i+2} = y - \delta\Sigma_1^{i+1}(x) - \delta f_{IF}^{i+1}(x,w,0) & = \delta_1; \\
(4) & \delta W_{i+3} = z - \delta\Sigma_1^{i+2}(y) - \delta f_{IF}^{i+2}(y,x,w) & = \delta_2; \\
(5) & \delta W_{i+4} = -w - \delta\Sigma_1^{i+3}(z) - \delta f_{IF}^{i+3}(z,y,x) & = 0; \\
(6) & \delta W_{i+5} = -x - \delta\Sigma_1^{i+4}(w) - \delta f_{IF}^{i+4}(w,z,y) & = 0; \\
(7) & \delta W_{i+6} = -y - \delta f_{IF}^{i+4}(0,w,z) & = 0; \\
(8) & \delta W_{i+7} = -z - \delta f_{IF}^{i+4}(0,0,w) & = u; \\
(9) & \delta W_{i+8} = & = -w.
\end{array}\right\} \quad (4.4.2)$$

The values of x, y and z from (4.2.1) are the following.

$$\begin{aligned}
x &= -\delta\Sigma_0^i(w) - \delta f_{MAJ}^i(w,0,0); \\
y &= -\delta f_{MAJ}^{i+1}(0,w,0); \\
z &= -\delta f_{MAJ}^{i+2}(0,0,w).
\end{aligned}$$

We now set conditions on the values for a and the e registers to obtain desired values for x, y and z and also to simplify the values of δWs. Using the kind of analysis done to obtain Rules 1 and 2, the following are easy to verify.

1. If $a_i = -1$ and $a_{i-1} = a_{i-2} = \alpha$, then $x = -1$.
2. If $a_{i+1} = a_{i-1}$, then $y = 0$; if $a_{i+1} = \overline{a_{i-1}}$, then $y = -1$.
3. If $a_{i+2} = a_{i+1}$, then $z = 0$; if $a_{i+2} = \overline{a_{i+1}}$, then $z = -1$.

Table 4.13: Values of a and e register for the δWs given by (4.4.1) to hold. The value of u is either 0 or w. We have $w = 1$, $x = -1$ and $z = 0$. If $y = 0$, then $\beta = \alpha$, while if $y = -1$, then $\beta = \overline{\alpha}$. By CDE, we have $\lambda = \beta + \alpha - \Sigma_0(\beta) - f_{MAJ}(\beta, -1, \alpha)$. Thus, the independent quantities are α, γ and μ.

index	$i-2$	$i-1$	i	$i+1$	$i+2$	$i+3$	$i+4$	$i+5$	$i+6$
a	α	α	-1	β	β				
e	γ	$\gamma+1$	-1	μ	λ	$\lambda+y$	-1	y	$-1-u$

Note. In our analysis of up to 22-round SHA-2, we saw that $z = 0$ arose as a necessary condition. Motivated by this, we will continue to work with $z = 0$. So, we will have $a_{i+2} = a_{i+1}$. Let this common value be β. Further, if $\beta = \alpha$, then $y = 0$ and if $\beta = \overline{\alpha}$, then $y = -1$. These and other values of a and e registers are shown in Table 4.13. We note the following.

1. If $y = 0$, then $\lambda = \alpha - \Sigma_0(\alpha)$.
2. If $y = -1$, then $\lambda = \alpha + \overline{\alpha} + 1 - \Sigma_0(\overline{\alpha}) = -\Sigma_0(\overline{\alpha})$.

At later point in the analysis, we will be obtaining λ and will require to obtain a corresponding value for α. In the case $y = -1$, $\alpha = \overline{\Sigma_0^{-1}(-\lambda)}$ and it is easy to obtain α from λ. The case $y = 0$ is not so simple. For SHA-256, one works with 32-bit words and then obtaining α from λ can be done by exhaustive search; however, for SHA-512, one has to work with 64-bit words and then things become more difficult. This is one of the reasons why it is more convenient to work with $y = -1$. (Note that $(w, x, y, z) = (1, -1, 0, 0)$ corresponds to the NB local collision, whereas $(w, x, y, z) = (1, -1, -1, 0)$ corresponds to the SS local collision.)

The values shown in Table 4.13 have been chosen so that the conditions on δW_{i+1} and δW_{i+5} to δW_{i+7} hold with probability one. Consider, for example, δW_{i+1}. From (4.4.2.2), we have

$$\begin{aligned}
\delta W_{i+1} &= x - \delta\Sigma_1^i(w) - \delta f_{IF}^i(w, 0, 0) \\
&= x - (\Sigma_1(e_i + w) - \Sigma_1(e_i)) \\
&\quad -(f_{IF}(e_i + w, e_{i-1}, e_{i-2}) - f_{IF}(e_i, e_{i-1}, e_{i-2})) \\
&= -1 - (0 - (-1)) - (e_{i-2} - e_{i-1}) \\
&= -2 - \gamma + \gamma + 1 \\
&= -1.
\end{aligned}$$

Similarly, Equations (4.4.2.6), (4.4.2.7) and (4.4.2.8) can be verified. Equations (4.4.2.3), (4.4.2.4) and (4.4.2.5) on the other hand give rise to the

following conditions on the values of α, γ and μ.

$$\left.\begin{array}{rcl} \delta_1 & = & y - \Sigma_1(\mu + x) + \Sigma_1(\mu) - f_{IF}(\mu + x, 0, \gamma + 1) \\ & & +f_{IF}(\mu, -1, \gamma + 1) \\ \delta_2 & = & -\Sigma_1(\lambda + y) + \Sigma_1(\lambda) - f_{IF}(\lambda + y, \mu + x, 0) \\ & & +f_{IF}(\lambda, \mu, -1) \\ w & = & -f_{IF}(\lambda + y, \lambda + y, \mu + x) + f_{IF}(\lambda + y, \lambda, \mu). \end{array}\right\} \quad (4.4.3)$$

The special case of these equations with $y = 0$ have been reported in [26] and a method for solving them has been discussed. The method to solve these equations is different for SHA-256 and for SHA-512. Next, we discuss methods to solve (4.4.3) for the case $y = -1$.

4.4.2 Solving Equation (4.4.3) for $y = -1$

The following provides an outline of the method to solve (4.4.3) for μ, γ and λ when $y = -1$ and δ_1 and δ_2 are given. From λ, we obtain α.

- The third equation holds with probability 1 if both λ and μ are odd.
- Given that λ and μ are odd, the second equation simplifies to $\delta_2 = -\Sigma_1(\lambda - 1) + \Sigma_1(\lambda) + \overline{(\lambda - 1)}$. For a given odd value of δ_2 occurring in the distribution of $\sigma_1(W) - \sigma_1(W - 1)$, it is possible to solve this equation for odd λ.
- Given such a λ, it is easy to solve the equation $\lambda = -\Sigma_0(\overline{\alpha})$ to obtain a suitable value of α, since Σ_0 is an invertible mapping for both SHA-256 and SHA-512.
- For the first equation, the term $-f_{IF}(\mu-1, 0, \gamma+1)+f_{IF}(\mu, -1, \gamma+1)$ is equal to μ, if γ is odd. This term is equal to $\mu-1$ if γ is even. Further, we note that $-\Sigma_1(\mu - 1) + \Sigma_1(\mu)$ is always even for both SHA-256 and SHA-512. Thus taking an arbitrary odd value of γ, the first equation is in the single variable μ and can be solved easily for a given δ_1.

Now we provide proofs of the observations above.

Lemma 4.4.1. *If $y = -1$, then the third equation of (4.4.3) is satisfied for any odd λ and odd μ.*

Proof. We have to show that

$$1 \;=\; -f_{IF}(\lambda - 1, \lambda - 1, \mu - 1) + f_{IF}(\lambda - 1, \lambda, \mu).$$

The quantities λ and $\lambda - 1$ differ only in their least significant bit since λ is odd. Similarly, μ and $\mu - 1$ differ only in their least significant bit since μ is odd. Let x_i denote the i^{th} bit of x, then λ_0=1, $(\lambda - 1)_0 = 0$, $\mu_0 = 1$ and

$(\mu-1)_0 = 0$. Let $(\lambda-1)_i = \lambda_i = 1$ and $(\lambda-1)_j = \lambda_j = 0$ for some non-zero indices i and j. Also, let $\mu_i = b_1$ and $\mu_j = b_2$ for these bit positions i and j. Now we are ready to write the bit patterns of the quantities occurring in the third equation.

bit	63	...	i ...	j	...	0
$\lambda-1$	.	...	1 ...	0	...	0
λ	.	...	1 ...	0	...	1
μ	.	...	b_1 ...	b_2	...	1
$f_{IF}(\lambda-1,\lambda,\mu)$	.	...	1 ...	b_2	...	1

Similarly,

bit	63	...	i ...	j	...	0
$\lambda-1$	.	...	1 ...	0	...	0
$\lambda-1$	.	...	1 ...	0	...	0
$\mu-1$	.	...	b_1 ...	b_2	...	0
$f_{IF}(\lambda-1,\lambda-1,\mu-1)$	.	...	1 ...	b_2	...	0

From the two bit patterns above, we get that

$$f_{IF}(\lambda-1,\lambda,\mu) - f_{IF}(\lambda-1,\lambda-1,\mu-1) = 1.$$

■

Lemma 4.4.2. *Let $y = -1$. For odd λ and odd μ, the second equation of (4.4.3) simplifies to $\delta_2 = -\Sigma_1(\lambda-1) + \Sigma_1(\lambda) + \overline{(\lambda-1)}$.*

Proof. Consider the following expression

$$-f_{IF}(\lambda-1,\mu-1,0) + f_{IF}(\lambda,\mu,-1).$$

Similar to the proof of the previous lemma, we consider the bit patterns of the quantities occurring in the above equation. Let $\lambda_i = 1$ and $\lambda_j = 0$ for some non-zero i, j. Also, let $\mu_i = b_1$ and $\mu_j = b_2$. Then the following bit patterns can be seen for the various quantities.

bit	63	...	i ...	j	...	0
λ	.	...	1 ...	0	...	1
μ	.	...	b_1 ...	b_2	...	1
-1	1	...	1 ...	1	...	1
$f_{IF}(\lambda,\mu,-1)$	.	...	b_1 ...	1	...	1

Similarly,

bit	63	...	i ...	j	...	0
$\lambda-1$	.	...	1 ...	0	...	0
$\mu-1$	.	...	b_1 ...	b_2	...	0
0	0	...	0 ...	0	...	0
$f_{IF}(\lambda-1,\mu-1,0)$	.	...	b_1 ...	0	...	0

Table 4.14: Values leading to collisions for different number of rounds of SHA-256. The value of i denotes the start point of the local collision, i.e., the local collision is placed from Round i to $i + 8$.

(#Rounds, i)	δ_1	δ_2	u	α	λ	γ	μ
(23, 8)	0	ff006001	0	32b308b2	051f9f7f	684e62b7	041fff81
(23, 9) (24, 10)	00006000	ff006001	1	32b308b2	051f9f7f	98e3923b	fbe05f81

From the two bit patterns above, we get that $f_{IF}(\lambda, \mu, -1)$ and $f_{IF}(\lambda - 1, \mu - 1, 0)$ will have the same bit value whenever the corresponding bit of λ is 1 and different bit value whenever the corresponding bit of λ is 0, except the least significant bit which will always be different. Comparing this difference with the bit pattern $\overline{\lambda - 1}$, we obtain

$$f_{IF}(\lambda, \mu, -1) - f_{IF}(\lambda - 1, \mu - 1, 0) = \overline{\lambda - 1}.$$

This completes the proof. ∎

Lemma 4.4.3. *Let $y = -1$. For odd μ and odd γ, the first equation of (4.4.3) simplifies to $\delta_1 = -1 - \Sigma_1(\mu - 1) + \Sigma_1(\mu) + \mu$.*

Proof. By considering the bit patterns of μ, $\mu - 1$ and $\gamma + 1$ the following can be proved in a manner similar to the previous two lemmas.

$$f_{IF}(\mu, -1, \gamma + 1) - f_{IF}(\mu - 1, 0, \gamma + 1) = \begin{cases} \mu & \text{if } \gamma \text{ is odd.} \\ \mu - 1 & \text{if } \gamma \text{ is even.} \end{cases}$$

Substituting the above value in the equation for δ_1 gives the required proof. ∎

SHA-256. For SHA-256 we did not solve the second equation explicitly since random search is itself good enough, producing a solution in few seconds. Solving all the three equations for α, γ and μ can be done in a few seconds on a current PC. Examples of values of (δ_1, δ_2) and the solutions to (4.4.3) for λ, γ and μ are provided in Table 4.14. The value of α is obtained from λ as explained earlier. The justification for choosing these particular values for the δs as well as the explanation for the first column will be provided later.

SHA-512. It is possible to solve (4.4.3) for SHA-512 as well, although we require a different approach than SHA-256. The main difference is in solving the first and the second equations. Since now 64-bit quantities are involved, it is no longer possible to solve the first and second equations by exhaustive

Table 4.15: Values leading to collisions for different number of rounds of SHA-512. The value of i denotes the start point of the local collision, i.e., the local collision is placed from Round i to $i+8$.

(# Rounds, i)	δ_1	δ_2	u	α	λ	γ	μ
(23, 8)	0	600000000237	0	7201b90f9f8df85e	3e000007ffdc9	1	43fffff800001
(23, 9) (24, 10)	200000000008	600000000237	1	7201b90f9f8df85e	3e000007ffdc9	1	45fffff800009

search. We describe a method to solve the second equation with the aid of an example.

Examples of values of (δ_1, δ_2) and the solutions to (4.4.3) for λ, γ and μ are provided in Table 4.15 and the value of α is obtained from λ as explained earlier. As in the case of SHA-256, the justification for choosing these particular values for the δs as well as the explanation for the first column will become clear later.

Solving the Second Equation of (4.4.3) For SHA-512. As shown in Lemma 4.4.2, for odd λ the second equation simplifies to

$$\delta_2 = -\Sigma_1(\lambda - 1) + \Sigma_1(\lambda) + \overline{(\lambda - 1)}.$$

We need to get an odd λ satisfying the above equation for a given value of δ_2. Since $-\Sigma_1(\lambda-1)+\Sigma_1(\lambda)$ is always even and $\overline{(\lambda-1)}$ is odd due to our choice of odd λ, we require δ_2 to be odd. This equation can be solved by hand. We explain the method to solve this equation for $\delta_2 = \mathsf{600000000237}$.

First note that $\Sigma_1(x)$ is the XOR addition of 3 n-bit quantities which are rotated/shifted forms of x. If λ is odd, then λ and $\lambda - 1$ differ only in the least significant bit. Therefore, the bit patterns of $\Sigma_1(\lambda)$ and $\Sigma_1(\lambda-1)$ will be same except at 3 bit positions. These 3 bit positions are indexed by 23, 46 and 50. By the structure of Σ_1 function and using the fact that λ is odd (i.e. $\lambda_0 = 1$), we have the following

$$\begin{aligned} b_1 = (\Sigma_1(\lambda))_{23} &= \lambda_0 \oplus \lambda_{37} \oplus \lambda_{41} = 1 \oplus \lambda_{37} \oplus \lambda_{41}, \\ b_2 = (\Sigma_1(\lambda))_{46} &= \lambda_0 \oplus \lambda_{23} \oplus \lambda_{60} = 1 \oplus \lambda_{23} \oplus \lambda_{60}, \\ b_3 = (\Sigma_1(\lambda))_{50} &= \lambda_0 \oplus \lambda_4 \oplus \lambda_{27} = 1 \oplus \lambda_4 \oplus \lambda_{27}. \end{aligned}$$

Also, because $(\lambda-1)_0 = 0$, we have $(\Sigma_1(\lambda-1))_{23} = \overline{b_1}$, $(\Sigma_1(\lambda-1))_{46} = \overline{b_2}$ and $(\Sigma_1(\lambda-1))_{60} = \overline{b_3}$.

Now consider the bit pattern of various quantities as follows.

bit	63	...	50	...	46	...	23	...	0
$A = \Sigma_1(\lambda - 1)$	.	...	$\overline{b_3}$	...	$\overline{b_2}$	...	$\overline{b_1}$	...	.
$B = \Sigma_1(\lambda)$	.	...	b_3	...	b_2	...	b_1	...	.
$A - B$	.	...	.	...	.	...	1	0...	0
δ_2	.	...	.	...	.	...	.	...	.
$A - B + \delta_2$	.	...	.	...	.	...	.	...	.

We require the quantity $(A - B + \delta_2)$ to be equal to $\overline{(\lambda - 1)}$. It is clear from the bit pattern above that the lowest 23 bits (indexed from 0 to 22) of $(A - B + \delta_2)$ will be same as those of δ_2. Equating these bits to corresponding bits of $\overline{(\lambda - 1)}$, we immediately get the lowest 23 bits of λ.

Now consider the bits between 23 and 46 of $(A - B)$. It is clear that all these bits will be equal. Further, all these bits will be equal to 1 if $b_1 = 1$ due to the borrow while subtracting B from A at bit position 23. Similarly, all these bits of $(A - B)$ will be equal to 0 if $b_1 = 0$. Our choice of δ_2 has all these bits equal to zero, hence the term $(A - B + \delta_2)$ will too have all these bits equal. But since this term is equal to $\overline{(\lambda - 1)}$, all these bits of $(\lambda - 1)$ will also be equal. Finally, note that λ and $(\lambda - 1)$ differ only in the lowest bit position, hence all the bits between 23 and 46 of λ will also be equal. In particular, we will have $\lambda_{37} = \lambda_{41}$, hence we have that $b_1 = 1 \oplus \lambda_{37} \oplus \lambda_{41} = 1$.

Continuing reasoning on bit positions in this way, for any given δ_2, either we can solve for λ or determine that a solution does not exist. For $\delta_2 =$ `600000000237` we obtained the solution $\lambda =$ `3e000007ffdc9`. Note that the method explained above does not require any particular structure of the bits of δ_2. As another example, we also solved for $\delta_2 =$ `19fffffffdd9` and obtained the solution as $\lambda =$ `2200000800227`.

Note.

1. The first equation can be solved in a similar manner for μ for a given δ_1.

2. It is possible to design an algorithm to do the task described above. But, such an algorithm will be complicated. Since we are interested in solving for a single value of δ_2, we chose not to describe and implement an algorithm. The method of solving by hand is good enough.

4.5 Finding 23 and 24-Round Collisions

We show that by suitably placing a local collision of the type described in Section 4.4.1 and using proper values for α, γ and μ, it is possible to obtain several 23 and 24-round collisions for SHA-2. For the description below, we will be considering the SS local collision, i.e., $(w, x, y, z) = (1, -1, -1, 0)$.

4.5.1 23-Round Collisions

There are two options of placing the SS local collision. From Round $i = 8$ to Round $i + 8 = 16$ and from Round $i = 9$ to Round $i + 8 = 17$. This gives rise to two kinds of 23-round collisions for SHA-2.

Case $i = 8$.

The local collision is started at $i = 8$ and ends at $i = 16$.

We have $(w, x, y, z) = (1, -1, -1, 0)$ and $\beta = \overline{\alpha}$. Also, we set $u = 0$ and $\delta_1 = 0$. We need to choose a suitable value for δ_2 which is the value of $\delta W_{i+3} = \delta W_{11}$. For this case, we let $\delta = \delta_2$. The value of δ_2 has to be chosen so that (4.4.3) has a solution. The time complexity of the algorithm depends on freq_δ (see Section 4.1.2 for the meaning of freq_δ) as explained below, so, one would like to choose δ such that freq_δ is as high as possible. At the same time, we have to ensure that (4.4.3) can be solved for the particular value of δ. Our choices of δ given in the rows with $(23, 8)$ of Tables 4.14 and 4.15 have the highest value of freq_δ for which it is possible to solve (4.4.3).

Since the local collision ends at Round 16, from Table 4.4.1 it necessarily follows that $\delta W_{16} = -1$. To obtain a 23-round collision, we want to ensure that $\delta W_{17} = \cdots = \delta W_{22} = 0$. From Table 4.8, (4.4.1) and the fact that $\delta W_j = 0$ for $0 \leq j \leq 7$, it follows that the condition $\delta W_{17} = \cdots = \delta W_{22} = 0$ is achieved if we can ensure $\delta W_{18} = 0$. Again, from Table 4.8, we have

$$\delta W_{18} = \delta\sigma_1(W_{16}) + \delta W_{11}. \tag{4.5.1}$$

So, for δW_{18} to be zero, we need $\delta = \delta W_{11} = -\delta\sigma_1(W_{16})$, so that δW_{11} should be one of the values which occur in the distribution of $\sigma_1(W) - \sigma_1(W - 1)$ for some W. (This is the reason why we analyzed the differential behavior of σ_1 in Section 4.1.2.) The word W_{16} is defined using message recursion and so, we cannot control this word directly. Instead, we analyze which message words can be used to control W_{16}.

First, let us consider which register values need to be set to specific values. Since $i = 8$, from Table 4.13, we see that a_6 to a_{10} and e_6 to e_{14} get defined. Using CDE, the value of e_{10} is actually determined by the values of a_6 to a_{10}. Using CDE, the values of e_9 down to e_6 determine the values of a_5 down to a_2. So, the values of a_2 to a_{10} and the values of e_{11} to e_{14} are fixed.

From message recursion, the expression for W_{16} is the following.

$$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0.$$

From the update function of the e-register, we have

$$W_{14} = e_{14} - (\Sigma_1(e_{13}) + f_{IF}(e_{13}, e_{12}, e_{11}) + a_{10} + e_{10} + K_{14}).$$

In this equation, all values other than W_{14} have already been fixed. So, W_{14} and hence $\sigma_1(W_{14})$ have fixed values. Let us now consider W_9. From the update function of the a-register, we can write

$$W_9 = a_9 - \Sigma_0(a_8) - f_{MAJ}(a_8, a_7, a_6) - \Sigma_1(e_8) - f_{IF}(e_8, e_7, e_6) - e_5 - K_9.$$

In the right hand side, all quantities other than e_5 have fixed values. Using CDE,

$$e_5 = a_5 + a_1 - \Sigma_0(a_4) - f_{MAJ}(a_4, a_3, a_2).$$

Again in the right hand side, all quantities other than a_1 have fixed values. So, we can write $W_9 = C - a_1$, where C is a fixed value. Now,

$$a_1 = \Sigma_0(a_0) + f_{MAJ}(a_0, b_0, c_0) + \Sigma_1(e_0) + f_{IF}(e_0, f_0, g_0) + h_0 + K_1 + W_1$$

where a_0 and e_0 depend on W_0 whereas b_0, c_0, f_0, g_0 and h_0 depend only on the initialization vector and hence are constants. Thus, we can write $a_1 = \Phi(W_0) + W_1$, where

$$\Phi(W_0) = \Sigma_0(a_0) + f_{MAJ}(a_0, b_0, c_0) + \Sigma_1(e_0) + f_{IF}(e_0, f_0, g_0) + h_0 + K_1.$$

We write $\Phi(W_0)$ to emphasize that this depends only on W_0. At this point, we can write

$$\begin{aligned} W_{16} &= \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0 \\ &= \sigma_1(W_{14}) + C - \Phi(W_0) - W_1 + \sigma_0(W_1) + W_0 \\ &= D - \Phi(W_0) - W_1 + \sigma_0(W_1) + W_0. \end{aligned} \tag{4.5.2}$$

We need to obtain W_0 and W_1 such that the value of W_{16} given by (4.5.2) satisfies the condition $\sigma_1(W_{16} - 1) - \sigma_1(W_{16}) = -\delta$ and then using (4.5.1) we obtain $\delta W_{18} = 0$ giving us the required condition of $\delta W_{17} = \cdots = \delta W_{22} = 0$.

Once W_0, W_1 have been obtained, a collision can be constructed in a manner similar to that for the 22-round case and as shown in Table 4.12. The idea is to first run SHA-2 for two rounds using W_0 and W_1. This determines the registers $(a_1, \ldots, h_1)$. Now, using Proposition 4.1.1, run SHA-2 step-by-step using W_i to set a_i to the desired value for $2 \leq i \leq 10$. Then run SHA-2 step-by-step using W_i to set e_i to the desired value for $11 \leq i \leq 14$. Finally, choose any value for W_{15}. The values of W'_i are determined by the values of W_i and δW_i for $0 \leq i \leq 15$. This gives a colliding message pair $(W_0, \ldots, W_{15})$ and $(W'_0, \ldots, W'_{15})$.

Estimate of Computation Effort. The main computational effort is in solving (4.5.2) for W_0 and W_1 such that $\sigma_1(W_{16} - 1) - \sigma_1(W_{16}) = -\delta$. We did not attempt an analytic solution. Instead, we tried random choices of W_0 and W_1 until we found a suitable W_{16}. There are freq_δ values of W_{16}

for which $\sigma(W_{16}) - \sigma(W_{16} - 1)$ equals δ. On an average, success is obtained after freq_δ trials. Each trial corresponds to about a single round of SHA-2 computation. So, the total cost of finding suitable W_0 and W_1 is about $\frac{\mathsf{freq}_\delta}{2^{4.5}}$ tries of 23-round SHA-2 computations.

SHA-256. The value of δ given in Table 4.14 is such that $\mathsf{freq}_\delta = 2^{16}$. (See Table 4.1 in Section 4.1.2.) So, the complexity of finding 23-round SHA-256 collision is about $2^{11.5}$ tries of 23-round SHA-256 computations. A message pair colliding for 23-round SHA-256 is given in Table A.20 of the Appendix.

SHA-512. In this case, we have estimates on freq_δ. (Again, see Section 4.1.2 for discussion on this issue.) For the particular value of δ given in Table 4.15, our estimate is $\mathsf{freq}_\delta \approx 2^{43}$. (See Table 4.2.) So, the effort required is about $\frac{\mathsf{freq}_\delta}{2^{4.5}} = \frac{2^{21}}{2^{4.5}} = 2^{16.5}$ trials of 23-round SHA-512. A message pair colliding for 23-round SHA-512 is given in Table A.23 of the Appendix.

Case $i = 9$.

It is possible to place the local collision from Round 9 to Round 17 and then perform an analysis to show that it is possible to obtain 23-round collisions for both $y = 0$ and $y = -1$. We do not provide these details, since a similar technique with an additional constraint is required for 24-round collision for which we provide complete details. An example of a collision obtained using this method is given in Table A.21 of the Appendix.

4.5.2 24-Round Collisions

The SS local collision is placed from Round $i = 10$ to Round $i + 8 = 18$, i.e. $(w, x, y, z) = (1, -1, -1, 0)$. The message differences are as given by (4.4.1) where we choose $u = 1$. The values of δ_1, δ_2 need to be suitably chosen and then the values of λ, γ and μ can be found by solving (4.4.3) as explained in Section 4.4.2. From λ, we find α as explained earlier.

Since the collision ends at Round 18 and $u = 1$, from (4.4.1) we have $\delta W_{17} = 1$ and $\delta W_{18} = -1$. To obtain a 24-round collision, we need to ensure $\delta W_{19} = \cdots = \delta W_{23} = 0$.

From Table 4.8, (4.4.1) and the fact that $\delta W_j = 0$ for $0 \leq j \leq 9$, we get that the conditions $\delta W_{19} = \delta W_{20} = 0$ translate into the conditions

$$\left.\begin{array}{rcl} \delta_1 = \delta W_{12} & = & -(\sigma_1(W_{17} + 1) - \sigma_1(W_{17})) \\ \delta_2 = \delta W_{13} & = & -(\sigma_1(W_{18} - 1) - \sigma_1(W_{18})). \end{array}\right\} \tag{4.5.3}$$

As in the case of 23-round collisions, based on the differential behavior of σ_1 (described in Section 4.1.2), we should try to choose δ_1 and δ_2 such that $\mathsf{freq}_{-\delta_1}$ and freq_{δ_2} are as high as possible.

Consider Table 4.13. This table tells us what the values of the different a and e-registers need to be. The values of a_8 to a_{12} and the values of e_8 to e_{16} get defined. Using CDE, the values of e_{11} down to e_8 determine the values of a_7 down to a_4. Thus, the values of a_4 to a_{12} and e_{13} to e_{16} are fixed. So, the values of a_0 to a_3 are free. In particular, we see that $e_{16} = -1 - u = -2$. This can be achieved by setting W_{16} to

$$W_{16} = e_{16} - \Sigma_1(e_{15}) - f_{IF}(e_{15}, e_{14}, e_{13}) - a_{12} - e_{12} - K_{16}. \quad (4.5.4)$$

Since all values on the right hand side are constants, we have that W_{16} is a constant value. On the other hand, W_{16} is defined by message recursion. So, we have to ensure that W_{16} takes the correct value. This is in addition to the requirement that the value of W_{17} and W_{18} satisfy (4.5.3).

We have already seen that W_{16} is a fixed value. Note that

$$\left.\begin{array}{rcl} W_{14} &=& e_{14} - \Sigma_1(e_{13}) - f_{IF}(e_{13}, e_{12}, e_{11}) - a_{10} - e_{10} - K_{14} \\ W_{15} &=& e_{15} - \Sigma_1(e_{14}) - f_{IF}(e_{14}, e_{13}, e_{12}) - a_{11} - e_{11} - K_{15}. \end{array}\right\} \quad (4.5.5)$$

Since for both equations, all the quantities on the right hand side are fixed values, so are W_{14} and W_{15}.

Using CDE twice, we can write

$$\left.\begin{array}{rcl} W_9 &=& -W_1 + C_4 + f_{MAJ}(a_4, a_3, a_2) - \Phi_0 \\ W_{10} &=& -W_2 + C_5 + f_{MAJ}(a_5, a_4, a_3) - \Phi_1 \\ W_{11} &=& -W_3 + C_6 + f_{MAJ}(a_6, a_5, a_4) - \Phi_2 \end{array}\right\} \quad (4.5.6)$$

where

$$\left.\begin{array}{rcl} C_i &=& e_{i+5} - \Sigma_1(e_{i+4}) - f_{IF}(e_{i+4}, e_{i+3}, e_{i+2}) - 2a_{i+1} \\ && -K_{i+5} + \Sigma_0(a_i) \\ \Phi_i &=& \Sigma_0(a_i) + f_{MAJ}(a_i, b_i, c_i) + \Sigma_1(e_i) + f_{IF}(e_i, f_i, g_i) \\ && +h_i + K_{i+1}. \end{array}\right\}$$

Using the expressions for W_9, W_{10} and W_{11} we obtain the following expressions for W_{16}, W_{17} and W_{18}.

$$\left.\begin{array}{rcl} W_{16} &=& \sigma_1(W_{14}) + C_4 - W_1 + f_{MAJ}(a_4, a_3, a_2) - \Phi_0 \\ && +\sigma_0(W_1) + W_0 \\ W_{17} &=& \sigma_1(W_{15}) + C_5 - W_2 + f_{MAJ}(a_5, a_4, a_3) - \Phi_1 \\ && +\sigma_0(W_2) + W_1 \\ W_{18} &=& \sigma_1(W_{16}) + C_6 - W_3 + f_{MAJ}(a_6, a_5, a_4) - \Phi_2 \\ && +\sigma_0(W_3) + W_2. \end{array}\right\} \quad (4.5.7)$$

We need to ensure that W_{16} has the desired value given by (4.5.4) and that W_{17} and W_{18} take values which satisfy (4.5.3).

The only free quantities are W_0 to W_3 which determine a_0 to a_3. The value of C_4 depends on e_8, e_7 and e_6, where e_8 has a fixed value and e_7 and e_6

are in turn determined using CDE by a_3 and a_2. Similarly, C_5 is determined by e_9, e_8 and e_7; where e_9, e_8 have fixed values and e_7 is determined using a_3. The value of C_6 on the other hand is fixed. Coming to the Φ values, Φ_0 is determined only by W_0; Φ_1 determined by W_0 and W_1; and Φ_2 determined by W_0, W_1 and W_2. Let

$$D = W_{16} - (\sigma_1(W_{14}) + C_4 + f_{MAJ}(a_4, a_3, a_2) - \Phi_0 + W_0). \quad (4.5.8)$$

If we fix W_0 and a_3, a_2, then the value of D gets fixed and we need to find W_1 such that the following equation holds.

$$D = -W_1 + \sigma_0(W_1). \quad (4.5.9)$$

A guess-then-verify algorithm can be used to solve this equation. This algorithm will be different for SHA-256 and for SHA-512 since the σ_0 function is different for the two. The guess-then-verify algorithms for both SHA-256 and SHA-512 are described in Section 4.5.3.

Solving (4.5.9) Using Table Look-Up. An alternative approach would be to use a pre-computed table. For each of the 2^n possible W_1s (n is the word size 32 or 64), prepare a table of entries $(W_1, -W_1 + \sigma_0(W_1))$ sorted on the second column. Then all solutions (if there are any) for (4.5.9) can be found by a simple look-up into the table using D. The table would have 2^n entries and if a proper index structure is used, then the look-up can be done very fast. We have not implemented this method.

Given $a_1, b_1, \ldots, h_1$ and a_2 the value of W_2 gets uniquely defined; similarly, given $a_2, b_2, \ldots, h_2$ and a_3, the value of W_3 gets uniquely defined. The equations are the following.

$$\left.\begin{array}{rcll} W_2 &=& a_2- & (\Sigma_0(a_1) + f_{MAJ}(a_1, b_1, c_1) + h_1 + \Sigma_1(e_1) \\ & & & +f_{IF}(e_1, f_1, g_1) + K_2) \\ W_3 &=& a_3- & (\Sigma_0(a_2) + f_{MAJ}(a_2, b_2, c_2) + h_2 + \Sigma_1(e_2) \\ & & & +f_{IF}(e_2, f_2, g_2) + K_3) \end{array}\right\} \quad (4.5.10)$$

The strategy for determining suitable $W_0, \ldots, W_3$ is the following.

1. Make random choices for W_0 and a_2, a_3.
2. Run SHA-2 with W_0 and determine Φ_0.
3. From a_3 and a_2 determine e_7 and e_6 using CDE.
4. Determine C_4 using (4.5.7) and then D using (4.5.8).
5. Solve (4.5.9) for W_1 using the guess-then-verify algorithm.
6. Run SHA-2 with W_1 to define $a_1, \ldots, h_1$.
7. Determine Φ_1 using (4.5.7) and then W_2 using (4.5.10).
8. Run SHA-2 with W_2 to define $a_2, \ldots, h_2$.
9. Determine Φ_2 using (4.5.7) and then W_3 using (4.5.10).

10. Compute W_{17} and W_{18} using (4.5.7).
11. If $\sigma_1(W_{17}+1) - \sigma_1(W_{17}) = -\delta_1$ and $\sigma_1(W_{18}-1) - \sigma_1(W_{18}) = \delta_2$, then return W_0, W_1, W_2 and W_3.

The values of W_0, W_1, W_2 and W_3 returned by this procedure ensure that the local collision ends properly at Round 18 and that $\delta W_j = 0$ for $j = 19, \ldots, 23$. This provides a 24-round collision. The actual construction of the collision is similar to the procedure for obtaining 22-round collisions described in Table 4.12; using the obtained values of $W_0, \ldots, W_3$ run SHA-2 for 4 rounds to define the values of $(a_3, \ldots, h_3)$. Use Proposition 4.1.1 to set $W_4, \ldots, W_{12}$ to values so that $a_4, \ldots, a_{12}$ get the required values. Set W_{13}, W_{14}, W_{15} to ensure that e_{13}, e_{14}, e_{15} get the required values. Finally, set $W'_i = W_i + \delta W_i$ for $i = 0, \ldots, 15$. Then the message pairs $(W_0, \ldots, W_{15})$ and $(W'_0, \ldots, W'_{15})$ provide a 24-round collision.

Estimate of Computation Effort. Let Round 5 involve a computation of g operations, where each operation is much faster than a single round of SHA-2; by our assessment the time for each operation is around 2^{-4} times the cost of a single round of SHA-2. Thus, the time for Round 5 is about $\frac{g}{2^4}$ single SHA-2 rounds. Further, let the success probability of the guess-then-verify attack be p. Then Round 5 needs to be repeated roughly $\frac{1}{p}$ times to obtain a solution.

By the choice of δ_1, the equality $\sigma_1(W_{17}+1) - \sigma_1(W_{17}) = -\delta_1$ holds roughly with probability $\frac{\mathsf{freq}_{\delta_1}}{2^n}$ while by the choice of δ_2 the equality $\sigma_1(W_{18}-1) - \sigma_1(W_{18}) = \delta_2$ holds roughly with probability $\frac{\mathsf{freq}_{\delta_2}}{2^n}$ and we obtain success in Round 11 with roughly $\frac{\mathsf{freq}_{\delta_1} \times \mathsf{freq}_{\delta_2}}{2^{2n}}$ probability. So, the entire procedure needs to be carried out around $\frac{2^{2n}}{\mathsf{freq}_{\delta_1} \times \mathsf{freq}_{\delta_2}}$ times to obtain a collision.

The guess-then-verify step takes about $g/2^4$ single SHA-2 rounds. The time for executing the entire procedure once is about $(\frac{g}{2^4} + 3)$ single SHA-2 rounds which is about $2^{-4.5} \times (\frac{g}{2^4} + 3)$ 24-round SHA-2 computations. Since the entire process needs to be repeated $\frac{2^{2n}}{\mathsf{freq}_{\delta_1} \times \mathsf{freq}_{\delta_2}}$ times for obtaining success, the number of 24-round SHA-2 computations till success is obtained is about

$$\left(\frac{2^{2n}}{\mathsf{freq}_{\delta_1} \times \mathsf{freq}_{\delta_2}}\right) \times \left(2^{-4.5} \times \left(\frac{g}{2^4} + 3\right) \times \frac{1}{p}\right).$$

If (4.5.9) is solved using a table look-up, then the cost estimate changes quite a lot. The cost of Round 5 reduces to about a single SHA-2 round so that the overall cost reduces to about

$$\left(\frac{2^{2n}}{\mathsf{freq}_{\delta_1} \times \mathsf{freq}_{\delta_2}}\right) \times \left(2^{-4.5} \times 3 \times \frac{1}{p}\right)$$

24-round SHA-2 computations. The trade-off is that we need to use a look-up table having 2^n entries.

SHA-256. We choose $\delta_2 =$ `ff006001` with $\mathsf{freq}_{\delta_2} = 2^{16}$. Also, we choose $\delta_1 =$ `00006000` so that $-\delta_1 =$ `ffffa000` and $\mathsf{freq}_{-\delta_1} = 2^{29} + 2^{26}$. (See Table 4.1 in Section 4.1.2.) (For choices of δ_2 with higher value of freq_{δ_2} there are no solutions to the second equation of (4.4.3).)

For these values of δ_1 and δ_2, it is possible to solve (4.4.3) to obtain suitable λ, γ and μ, which in turn determine α. An example of these values is shown in Table 4.14 in the row $(24, 9)$. (The same values also hold for obtaining 23-round collision by placing a local collision from Round 9 to 17.)

The values of g, freq_{δ_1} and freq_{δ_2} are 2^{18}, 2^{29} and 2^{16} respectively. So, the time complexity is about $2^{28.5}$ 24-round SHA-256 computations. In our experiments, we found that the computation effort required to find $W_0, \ldots, W_3$ actually turns out to be less than the estimated effort of $2^{28.5}$ 24-round SHA-256 computations. The value of $2^{28.5}$ matches the figure given in [26], but [26] does not provide the detailed analysis of their cost. A message pair colliding for 24-round SHA-256 is given in Table A.22 of the Appendix.

As already explained, if (4.5.9) is solved using a table look-up, then the cost reduces to about $2^{15.5}$ 24-round SHA-256 computations.

SHA-512. We choose $\delta_2 =$ `600000000237` with $\mathsf{freq}_{\delta_2} \approx 2^{43}$. Also, we choose $\delta_1 =$ `200000000008` so that $\mathsf{freq}_{-\delta_1} \approx 2^{61.5}$. See Table 4.2 in Section 4.1.2 For these values of δ_1 and δ_2, it is possible to solve (4.4.3) to obtain suitable λ, γ and μ, which in turn determine α. An example of these values is shown in the row marked $(24, 10)$ of Table 4.15.

The guess-then-verify attack for SHA-512 case requires $g = 2^{15}$ operations. Hence, the effort required for 24-round SHA-512 attack is about

$$\left(\frac{2^{2\times 64}}{2^{61.5} \times 2^{43}}\right) \times \left(2^{-4.5} \times \left(\frac{2^{15}}{2^4} + 3\right) \times \frac{1}{2^{-2.5}}\right) = 2^{32.5}$$

trials of 24-round SHA-512. In [26], the corresponding effort is 2^{53} trials of 24-round SHA-512. This significant improvement in the attack complexity allows us to provide the first example of a colliding message pair for 24-round SHA-512. A message pair colliding for 24-round SHA-512 is given in Table A.24 of the Appendix.

Note that using a table having 2^{64} entries to solve (4.5.9) will reduce the computational effort to about $2^{22.5}$ trials of 24-round SHA-512.

4.5.3 Guess-Then-Verify Algorithm for Solving Equation (4.5.9)

For the ease of notation, in this section we will use W instead of W_1.

For SHA-256.

Consider Table 4.1 where the structure of W and $\sigma_0(W)$ is shown for SHA-256. We have $-W+\sigma_0(W) = D$, where $D = (d_{31}, \ldots, d_0)$ is a 32-bit constant. For $31 \geq k \geq l \geq 0$, we will use the notation $X[k, l]$ to denote bits $x_k, \ldots, x_l$ of the 32-bit quantity X.

We explain how the guess-then-verify algorithm proceeds. Suppose that we guess $W[14, 0]$. Let $X = D+W$ and $Y = (W[14, 0] \gg 3) \oplus (W[14, 0] \gg 7)$. Then $W[25, 18] = (X \oplus Y)\&(\texttt{ff})$. Having determined $W[25, 18]$ we next determine $W[29, 26]$ using positions 22 to 19 of Table 4.1. This time, however, there may have been a possible carry into the 19th bit and we need to account for that. Let c_0 be a bit. Define $X = (D \gg 19) + (W[25, 18] \gg 1) + c_0$ and $Y = (W[14, 0] \gg 5) \oplus (W[25, 18] \gg 4)$. Then $W[29, 26] = (X \oplus Y)\&(\texttt{f})$. This illustrates the general idea and can be extended to determine the other bits. Once the entire W has been determined we need to determine whether $-W + \sigma_0(W) = D$. The entire algorithm is shown in Figure 4.2. This algorithm involves guessing $W[14, 0]$ and bits c_0, c_1, c_2, which is a total of 18 bits. If the equation $D = -W+\sigma_0(W)$ does not have any solution, then none will be returned by this algorithm; on the other hand, if there is a solution or there are more than one solutions, then all solutions will be returned. A total of 2^{18} operations are required. The time for each operation is significantly less than the time for a single SHA-256 round and by our assessment it is about 2^{-4} times the time for a single SHA-256 round.

Note. In [26], it has been remarked that "*by guessing the least 15 bits of W_1 the entire W_1 can be reconstructed and with probability 2^{-14} it is going to be correct*". No details are provided. In particular, the guess-then-verify algorithm that we have described is not present in [26].

In our experiments with SHA-256, we found that for almost every other value of D, (4.5.9) has solutions, the number of solutions being one or two. So, for a random choice of D, we consider (4.5.9) to hold with probability $p \approx 1$.

For SHA-512.

Consider Table 4.3 where the structure of W and $\sigma_0(W)$ is shown for SHA-512. We have $-W+\sigma_0(W) = D$, where $D = (d_{63}, \ldots, d_0)$ is a 64-bit constant. For $63 \geq k \geq l \geq 0$, we will use the notation $X[k, l]$ to denote bits $x_k, \ldots, x_l$ of the 64-bit quantity X.

We explain how the guess-then-verify attack proceeds. Suppose that we guess $W[7, 0]$. So we know the 7 bits $W[7, 1]$ and $W[6, 0]$. Now, consider the lowest 7 bits of $D + W$. We need $D + W$ to be equal to $\sigma_0(W)$. The term $\sigma_0(W)$ consists of 3 quantities XOR'ed, one of which, $W[7, 1]$, is already known. The other two quantities are $W[13, 7]$ and $W[14, 8]$. So we can

Figure 4.1: Structure of W and $\sigma_0(W)$ for SHA-256.

W	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}	w_{24}
$W \gg 3$	0	0	0	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}
$W \ggg 7$	w_6	w_5	w_4	w_3	w_2	w_1	w_0	w_{31}
$W \ggg 18$	w_{17}	w_{16}	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}
W	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}	w_{16}
$W \gg 3$	w_{26}	w_{25}	w_{24}	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}
$W \ggg 7$	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}	w_{24}	w_{23}
$W \ggg 18$	w_9	w_8	w_7	w_6	w_5	w_4	w_3	w_2
W	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9	w_8
$W \gg 3$	w_{18}	w_{17}	w_{16}	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}
$W \ggg 7$	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}	w_{16}	w_{15}
$W \ggg 18$	w_1	w_0	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}
W	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0
$W \gg 3$	w_{10}	w_9	w_8	w_7	w_6	w_5	w_4	w_3
$W \ggg 7$	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9	w_8	w_7
$W \ggg 18$	w_{25}	w_{24}	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}

Figure 4.2: A guess-then-verify algorithm for solving $D = -W + \sigma_0(W)$ for SHA-256.

1. Guess $W[14, 0]$.
2. Let $X = D + W$ and $Y = (W[14, 0] \gg 3) \oplus (W[14, 0]) \gg 7$
   and set $W[25, 18] = (X \oplus Y)\&(\texttt{ff})$.
3. Guess c_0.
4. Let $X = (D \gg 19) + (W[25, 18] \gg 1) + c_0$,
   and $Y = (W[14, 0] \gg 5) \oplus (W[25, 18] \gg 4)$
   and set $W[29, 26] = (X \oplus Y)\&(\texttt{f})$.
5. Guess c_1.
6. Let $X = (D \gg 23) + (W[25, 18] \gg 6) + c_1$
   and $Y = (W[14, 0] \gg 9) \oplus (W[29, 26] \gg 4)$,
   and set $W[31, 20] = (X \oplus Y)\&(\texttt{3})$.
7. Guess c_2.
8. Let $X = (D \gg 8) + (W[14, 0] \gg 8) + c_2$
   and $Y = (W[14, 0] \gg 11) \oplus (W[29, 26])$,
   and set $W[31, 20] = (X \oplus Y)\&(\texttt{7})$.
9. If $-W + \sigma_0(W) = D$, then output W as one solution.

compute $X = W[13,7] \oplus W[14,8] = (D+W) \oplus W[7,1]$. Now, consider the least significant bit of X. This is the XOR of $W[7]$ and $W[8]$. We already know $W[7]$, so it is possible to compute $W[8]$. Once $W[8]$ is known, we can compute $W[9]$ by considering the second least significant bit of X. Continuing this way, we can get $W[14,7]$.

Now consider the quantity $(D+W) \oplus (W \ggg 1)$ for bit positions 7 to 13. If the possible carry bit into the addition $D+W$ at bit position 7 can be guessed, then $W[21,15]$ can be determined. Extending this reasoning further, we need to guess 7 carry bits and the initial 8 bits of W to completely determine W. If the obtained value of W satisfies $-W + \sigma_0(W) = D$, then we have the correct solution. The entire algorithm is shown in Figure 4.4.

In the algorithm, we use a function GTD, which takes low order $7i$ bits of W as input and produces low order $7i+7$ bits of W. This function is described at the end of the figure.

This algorithm involves guessing $W[7,0]$ and bits $c_1, c_2, \ldots c_7$, which is a total of 15 bits. If the equation $D = -W + \sigma_0(W)$ does not have any solution, then none will be returned by this algorithm; on the other hand, if there is a solution or there are more than one solutions, then all solutions will be returned. A total of 2^{15} operations are required. The time for each operation is significantly less than the time for a single SHA-512 round and by our assessment it is about 2^{-4} times the time for a single SHA-512 round.

4.6 A Property of the NB Local Collision for SHA-512

The NB local collision has $(w,x,y,z) = (1,-1,0,0)$; $\delta W_i = 1$, $\delta W_{i+8} = -1$ and $\delta W_j = 0$, for $j = i+4, i+5, i+6, i+7$. Here $8 \leq i \leq 10$. The message word differences $\delta W_{i+1}, \delta W_{i+2}$ and δW_{i+3} are given by the following equations:

$$\left.\begin{array}{rcl} \delta W_{i+1} &=& -1 - \delta f_{IF}^{i}(1,0,0) - \delta\Sigma_1(e_i), \\ \delta W_{i+2} &=& -\delta f_{IF}^{i+1}(-1,1,0) - \delta\Sigma_1(e_{i+1}), \\ \delta W_{i+3} &=& -\delta f_{IF}^{i+2}(0,-1,1). \end{array}\right\} \tag{4.6.1}$$

Suppose that the NB local collision is placed between Round i and Round $i+8$, $i = 8, 9, 10$; and it is desired to obtain a collision for $i+14$ rounds. Since the local collision ends at Round $i+8$, from the differential path of the local collision in Table 4.3, we require the difference in the message word δW_{i+8} to be -1.

The basic idea is to ensure that the message word differences are all zero after the local collision ends. This will ensure that the two messages will not introduce any difference in the registers. Therefore, $\delta W_{i+9} = \delta W_{i+10} = \ldots =$

Figure 4.3: Structure of W and $\sigma_0(W)$ for SHA-512.

W	w_{63}	w_{62}	w_{61}	w_{60}	w_{59}	w_{58}	w_{57}	w_{56}
$W \gg 7$	0	0	0	0	0	0	0	w_{63}
$W \ggg 1$	w_0	w_{63}	w_{62}	w_{61}	w_{60}	w_{59}	w_{58}	w_{57}
$W \ggg 8$	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0
W	w_{55}	w_{54}	w_{53}	w_{52}	w_{51}	w_{50}	w_{49}	w_{48}
$W \gg 7$	w_{62}	w_{61}	w_{60}	w_{59}	w_{58}	w_{57}	w_{56}	w_{55}
$W \ggg 1$	w_{56}	w_{55}	w_{54}	w_{53}	w_{52}	w_{51}	w_{50}	w_{49}
$W \ggg 8$	w_{63}	w_{62}	w_{61}	w_{60}	w_{59}	w_{58}	w_{57}	w_{56}
W	w_{47}	w_{46}	w_{45}	w_{44}	w_{43}	w_{42}	w_{41}	w_{40}
$W \gg 7$	w_{54}	w_{53}	w_{52}	w_{51}	w_{50}	w_{49}	w_{48}	w_{47}
$W \ggg 1$	w_{48}	w_{47}	w_{46}	w_{45}	w_{44}	w_{43}	w_{42}	w_{41}
$W \ggg 8$	w_{55}	w_{54}	w_{53}	w_{52}	w_{51}	w_{50}	w_{49}	w_{48}
W	w_{39}	w_{38}	w_{37}	w_{36}	w_{35}	w_{34}	w_{33}	w_{32}
$W \gg 7$	w_{46}	w_{45}	w_{44}	w_{43}	w_{42}	w_{41}	w_{40}	w_{39}
$W \ggg 1$	w_{40}	w_{39}	w_{38}	w_{37}	w_{36}	w_{35}	w_{34}	w_{33}
$W \ggg 8$	w_{47}	w_{46}	w_{45}	w_{44}	w_{43}	w_{42}	w_{41}	w_{40}
W	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}	w_{24}
$W \gg 7$	w_{38}	w_{37}	w_{36}	w_{35}	w_{34}	w_{33}	w_{32}	w_{31}
$W \ggg 1$	w_{32}	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}
$W \ggg 8$	w_{39}	w_{38}	w_{37}	w_{36}	w_{35}	w_{34}	w_{33}	w_{32}
W	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}	w_{16}
$W \gg 7$	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}	w_{24}	w_{23}
$W \ggg 1$	w_{24}	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}
$W \ggg 8$	w_{31}	w_{30}	w_{29}	w_{28}	w_{27}	w_{26}	w_{25}	w_{24}
W	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9	w_8
$W \gg 7$	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}	w_{16}	w_{15}
$W \ggg 1$	w_{16}	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9
$W \ggg 8$	w_{23}	w_{22}	w_{21}	w_{20}	w_{19}	w_{18}	w_{17}	w_{16}
W	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0
$W \gg 7$	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9	w_8	w_7
$W \ggg 1$	w_8	w_7	w_6	w_5	w_4	w_3	w_2	w_1
$W \ggg 8$	w_{15}	w_{14}	w_{13}	w_{12}	w_{11}	w_{10}	w_9	w_8

Figure 4.4: A guess-then-verify algorithm for solving $D = -W + \sigma_0(W)$ for SHA-512.

1. Guess $W[7, 0]$ and carry bits $c_1, c_2, c_3, c_4, c_5, c_6, c_7$.
2. Let $c_0 = 0$.
3. for($i = 0$; $i \leq 7$; i++)
4. $\quad W[7i + 7, 0] = \text{GTD}(W[7i, 7i - 6], c_i)$;
5. If $-W + \sigma_0(W) = D$, then output W as one solution.

1. function GTD($W[7i, 7i - 6], c_i$){
2. $X = (((D \gg (7i - 7))\&(\texttt{7f})) + c_i + (W \gg (7i - 7))\&(\texttt{7f}))$
$\qquad \oplus((W \gg (7i - 6))\&(\texttt{7f}))$;
3. $\quad T_1 = (X\&\texttt{1}) \oplus ((W \gg (7i))\&\texttt{1})$;
4. $\quad T_2 = ((X \gg 1)\&\texttt{1}) \oplus T1$;
5. $\quad T_3 = ((X \gg 2)\&\texttt{1}) \oplus T2$;
6. $\quad T_4 = ((X \gg 3)\&\texttt{1}) \oplus T3$;
7. $\quad T_5 = ((X \gg 4)\&\texttt{1}) \oplus T4$;
8. $\quad T_6 = ((X \gg 5)\&\texttt{1}) \oplus T5$;
9. $\quad T_7 = ((X \gg 6)\&\texttt{1}) \oplus T6$;
10. temp $= T_1 \oplus (T_2 \ll 1) \oplus (T_3 \ll 2) \oplus (T_4 \ll 3) \oplus (T_5 \ll 4)$
$\qquad \oplus(T_6 \ll 5) \oplus (T_7 \ll 6)$;
11. $W[7i + 7, 0] = W[7i, 7i - 6] \oplus (temp \ll (7i + 1))$;
12. Return $W[7i + 7, 0]$ }.

$\delta W_{i+14} = 0$. From Table 4.8 it follows that we require $\delta\sigma_1(\delta W_{i+8}) + \delta W_{i+3} = 0$ to ensure that $\delta W_{i+10} = 0$.

We now show for SHA-512, it is difficult to find values of δW_{i+3} and $\delta\sigma_1(W_{i+8})$ which are of the same order of magnitude. The values of δW_{i+3} are biased towards small magnitudes. In contrast, the values of $\sigma_1(W_{i+8}) - \sigma_1(W_{i+8}-1)$ for SHA-512 are biased towards large magnitudes. This makes it difficult to achieve equality of the two terms as required to ensure $\delta W_{i+10} = 0$.

In the discussion that follows, we use X_i to denote the i^{th} bit of a 64-bit quantity X. We also use the convention that the index of the least significant bit is 0.

Proposition 4.6.1. *$Pr[P_j \neq (P+1)_j] = 1/2^j$, where the probability is taken over random P.*

Proof. The necessary and sufficient condition for the j^{th} bit of P and $P+1$ to differ is that all the bits from 0 to $(j-1)$ in P are 1. This happens with probability $1/2^j$, hence proved. ∎

Proposition 4.6.2. *If two numbers X and Y are such that $X_i \neq Y_i$ and $X_{i-1} = Y_{i-1}$, then $|X - Y| \geq 2^{i-1} + 1$.*

Proof. Without loss of generality, suppose $X_i = 1$ and $Y_i = 0$. Let $Z = X - Y$. If $Z_i = 1$, then clearly $Z \geq 2^i$ and we are done.

So, suppose $Z_i = 0$ and consider the process of binary subtraction of Y from X to obtain Z. Since $X_i = 1$ and $Y_i = 0$, the result $Z_i = 0$ can happen only if the subtraction of $Y_{i-1}Y_{i-2}\ldots Y_0$ from $X_{i-1}X_{i-2}\ldots X_0$ produces a carry. But since $X_{i-1} = Y_{i-1}$, this implies the following two things.

1. $Z_{i-1} = 1$.
2. The subtraction of $Y_{i-2}Y_{i-3}\ldots Y_0$ from $X_{i-2}X_{i-3}\ldots X_0$ produces a carry.

    1. $Z_{i-1} = 1$.
    2. The subtraction of $Y_{i-2}Y_{i-3}\ldots Y_0$ from $X_{i-2}X_{i-3}\ldots X_0$ produces a carry.

The second point implies that at least one bit of $Z_{i-2}Z_{i-3}\ldots Z_0$ must be 1. This together with the first point $Z_{i-1} = 1$ implies that $Z \geq 2^{i-1} + 1$. Hence proved. ∎

Next we prove that the probability that the absolute value of δW_{i+3}, in the NB local collision is larger than 2^j is bounded above by $1/2^{j-1}$.

Lemma 4.6.3. *If the NB local collision is started at Round i, then*

$$Pr[|\delta W_{i+3}| \geq 2^j] < 1/2^{j-1}.$$

Proof. Since the local collision is started from round i, the message difference δW_{i+3} is given by Equation 4.6.1. This equation gives:

$$\begin{aligned}\delta W_{i+3} &= -\delta f_{IF}^{i+2}(0,-1,1),\\ &= -f_{IF}(e_{i+2}, f_{i+2}-1, g_{i+2}+1) + f_{IF}(e_{i+2}, f_{i+2}, g_{i+2}),\\ &= -f_{IF}(e_{i+2}, e_{i+1}-1, e_i+1) + f_{IF}(e_{i+2}, e_{i+1}, e_i).\end{aligned}$$

The two f_{IF} terms in the computation above have the same first argument e_{i+2}. The second and the third arguments have a modular difference of ± 1. If the j^{th} bit of e_{i+2} is 1 then the two f_{IF} functions will select the corresponding bit from the middle argument, else from the third argument.

Let $A = f_{IF}(e_{i+2}, e_{i+1}-1, e_i+1)$ and $B = f_{IF}(e_{i+2}, e_{i+1}, e_i)$. Further, let P_n be the event that $A_n \neq B_n$. The event $\delta W_{i+3} \geq 2^j$ can happen if and only if at least one of the bits $j, j+1, \ldots 63$ of δW_{i+3} is 1, i.e., if and only if at least one of the events $P_j, P_{j+1}, \ldots P_{63}$ holds.

Now we are ready to bound the probability of the required event. In the fourth step below, we use the fact that $f_{IF}(a,b,c) = b$ if $a = 1$ and $= c$ if $a = 0$.

$$\begin{aligned}Pr[\delta W_{i+3} \geq 2^j] &= Pr[\bigcup_{i\geq j} P_i]\\ &\leq \sum_{i\geq j} Pr[P_i]\\ &= \sum_{i\geq j} (Pr[(e_{i+2})_i = 0] \cdot Pr[P_i|((e_{i+2})_i = 0)]\\ &\qquad + Pr[(e_{i+2})_i = 1] \cdot Pr[P_i|((e_{i+2})_i = 1)])\\ &= \sum_{i\geq j} \left(\frac{1}{2} \cdot Pr[(e_i+1)_i \neq e_i]\right.\\ &\qquad \left. + \frac{1}{2} \cdot Pr[(e_{i+1}-1)_i \neq e_{i+1}]\right)\\ &= \frac{1}{2} \cdot \sum_{i\geq j} \left(\frac{1}{2^i} + \frac{1}{2^i}\right) \quad \text{(Using Proposition 4.6.1)}\\ &< \frac{1}{2^{j-1}}.\end{aligned}$$

This proves the result. ∎

We now look at the distribution of values of $\sigma_1(W) - \sigma_1(W-1)$ for random choices of W.

Lemma 4.6.4. *For the function σ_1 used in SHA-512,*

$$|\sigma_1(W) - \sigma_1(W-1)| \geq (2^{42} + 2^{39} + 2^{38} + 2^{36} - 2^3),$$

where W is any 64-bit word.

Proof. The function σ_1 is defined for SHA-512 as:

$$\sigma_1(W) = ROTR^{19}(W) \oplus ROTR^{61}(W) \oplus SHR^6(W). \tag{4.6.2}$$

Let the 64-bit word W be specified as $(w_{63}, w_{62}, \ldots, w_1, w_0)$ where w_0 is the least significant bit of W. Then $\sigma_1(W)$ can be expressed as bit-wise XOR of three quantities having bit pattern shown below.

Bit Index	63	62	...	58	57	...	45	44	...	0
$ROTR^{19}$	w_{18}	w_{17}	...	w_{13}	w_{12}	...	w_0	w_{63}	...	w_{19}
$ROTR^{61}$	w_{60}	w_{59}	...	w_{55}	w_{54}	...	w_{42}	w_{41}	...	w_{61}
SHR^6	0	0	...	0	w_{63}	...	w_{51}	w_{50}	...	w_6

Let $W' = W - 1$. Then similar structure for $\sigma_1(W')$ can also be visualized. We are interested in the magnitude of $\sigma_1(W) - \sigma_1(W')$.

Let j be the least index such j^{th} bit of W is 1. That is, $w_j = 1$ and $w_i = 0$ for all $i \leq j - 1$. Then we have, $w_i \neq w'_i$ for $i \leq j$ and $w_i = w'_i$ for $i > j$. Now we consider two cases for j.

Case 1: $0 \leq j \leq 40$. In this case, we have that $w_i = w'_i$ for $i = 63$, 51, 50, 42, 41 and $w_0 \neq w'_0$. From the structure of $\sigma_1(W)$ and $\sigma_1(W')$, we note that their 45^{th} bits will be unequal but their 44^{th} bits will be equal. Using Proposition 4.6.2 we get, $|\sigma_1(W) - \sigma_1(W')| \geq 2^{44} + 1$.

Case 2: $j \geq 41$. We need to consider the individual cases $j = 41, 42, \ldots 63$ here. Consider the case $j = 41$ first. In this case, we know the exact bit pattern in W and W' up to 41 bits. Only the high order bits from 42 to 63 are unknown in these two quantities. We also know that these high order bits are the same in W and W'. Since these are only 22 bits, we can exhaustively search this space and compute the value $|\sigma_1(W) - \sigma_1(W')|$ for the case $j = 41$. As j is increased, the same idea can be used with even smaller search space. The size of the complete search space is $1 + 2 + 2^2 + \ldots + 2^{22} = 2^{23} - 1$. A C program running on an ordinary PC takes a fraction of a second to traverse this space.

Using exhaustive search, we found the minimum vale of $|\sigma_1(W) - \sigma_1(W - 1)|$ to be `000004cfffffff8` which occurred for $j = 42$. This value is equal to $(2^{42} + 2^{39} + 2^{38} + 2^{36} - 2^3)$.

We have left one particular case of W undiscussed. This is the special case when all the bits in W are zero. In this case, we can compute the difference directly since $\sigma_1(0) = 0$ and $\sigma_1(-1) = 1 + 2 + \ldots + 2^{57}$. Thus, we have the difference $= 2^{58} - 1$.

Combining all the cases, the result is proved. ∎

4.7 Extending the Attack to Longer Rounds

The method of attack described so far cannot be meaningfully extended beyond 24 rounds as already mentioned in [26]. This is due to the fact that every extra round will introduce a new condition on the previous message words. The 24-round collision already utilized the freedom in the first message word W_0. To have a 25-round collision by starting the local collision at Round $i = 11$, will introduce impossibility in ensuring that the message word difference $\delta W_{16} = 0$. This is explained below.

As shown in Section 4.4.1, the local collision is $\{w, -w, \delta_1, \delta_2, 0, 0, 0, u, w\}$. If we start this local collision at Round $i = 11$, then $\delta W_{15} = \delta W_{16} = \delta W_{17} = 0$. Now from the message recursion of SHA-2, we have:

$$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0.$$

All the terms in the above equation, except W_{14}, are zero. Therefore this equation cannot be satisfied by this local collision. Similar reasons apply for longer round collisions.

Perhaps more fundamentally the problem is that, we are using only a single local collision. Since the local collision is nonlinear in nature, it is difficult to combine two or more such collisions. Further progress in analysis of round-reduced SHA-256 collisions will require some method to combined more than one (linear or non-linear) local collision.

4.8 Conclusions

In this chapter we studied recent nonlinear attacks against the SHA-2 family. We developed new attacks against up to 24-round SHA-256 and up to 24-round SHA-512. We have utilized some weaknesses of the SHA-2 compression function to construct these attacks. In the next chapter, we attempt to make the SHA-2 family resilient against these attacks.

Chapter 5

A New Hash Family Obtained by Modifying the SHA-2 Family

5.1 Introduction

In this chapter, we study several properties of the SHA-2 design which have been utilized in recent collision attacks against reduced round SHA-2. Small modifications to the SHA-2 design are suggested to thwart these attacks. The modified round function provides the same resistance to linearization attacks as the original SHA-2 round function, but provides better resistance to non-linear attacks. Our next contribution is to introduce the general idea of "multiple feed-forward" for the construction of cryptographic hash functions. This can provide increased resistance to the Chabaud-Joux type "perturbation-correction" collision attacks. The idea of feed-forward is taken further by introducing the idea of feed-forward across message blocks leading to resistance against generic multi-collision attacks. The net effect of the suggested changes to the SHA-2 design has insignificant impact on the efficiency of computing the digest.

In the previous chapters we studied linear and non-linear local collisions and developed attacks against up to 24-round SHA-2 using these local collisions. Recent attacks starting with [38] and followed by [26, 49–51] utilized certain previously unknown properties in the round function of SHA-2. While none of these attacks threaten any of the security properties of the full SHA-2 hash functions, it is also true that some features of the compression function have been exploited in the attack. The following four properties have been used in the recent round-reduced attacks.

1. The works in [22], [24], [34, 35] and [47] show that 9-round local collisions obtained using XOR differentials hold with probability 2^{-39} or less

for SHA-256. In contrast, the work of [38] shows that 9-round local collisions using additive differentials can be obtained with probability 1/3 (and improved to probability 1 in [51]). This suggests that the round function resists XOR differentials better than additive differentials.

2. SHA-2 design uses 8 registers a to h, where a and e registers are non-linearly updated while the rest are simply copied. It turns out that there is a *very simple* relation by which the e-register value at Round i can be controlled using only the a-register values at Round i to $i-4$.

3. The round update function for the a-register uses an invertible linear transformation Σ_0 while that of the e-register uses an invertible linear transformation Σ_1. Both Σ_0 and Σ_1 have 0 and -1 as fixed points.

4. The technique of perturbation-correction [5] is used to build the attacks. A 9-round local collision is suitably placed between rounds i and $i+8$ and it is ensured that all message word differences after Round $i+8$ are zero.

We suggest methods to get around the above issues. The linear maps Σ_0, Σ_1 are modified to affine maps Γ_0, Γ_1 to ensure that Γ_0, Γ_1 and $\Gamma_0 \oplus \Gamma_1$ do not have any fixed points (along with a few other properties). This takes care of the third point above. Simple but carefully considered changes are suggested to the update function of the a and e-registers. These changes help in avoiding local collisions of high probability based on additive differentials which have been obtained in [38] and [51]. These changes also lead to cancelling out the simple relation between the a and e-registers. As a result, both the first and second points above are eliminated.

The resistance of the new round function to linearization attacks is the same as that provided by the SHA-2 round function. In particular, the best linear local collision for the 256-bit case holds with probability 2^{-39} as in the case of SHA-256. We argue later that the suggested changes provide better resistance to recent non-linear attacks.

To tackle the fourth point, we introduce a new hash function design called multiple feed-forward. This provides additional resistance to the perturbation-correction technique for finding collisions. The SHA-family design uses a single feed-forward where the chaining value is added to the output of composition of all the round functions. We suggest introducing several other feed-forward rounds where the feed-forward is alternately provided using addition and XOR. A consequence is that if any 9-round local collision is placed within the first 16 rounds, then there will be a round i within these 16 rounds such that there will be a perturbation in the registers at Round i and this perturbation will necessarily extend to rounds beyond the first 16 rounds. Since message words beyond the first 16 rounds are obtained using

the message recursion, it will be difficult to cancel out the effect of such cascaded perturbation. This improves the resistance to perturbation-correction attacks. Such resistance is achieved at a marginal cost. The amortized cost of the new feed-forward steps is less than one t-bit operation (add/XOR) per round, where $t = 32$ for SHA-256 and $t = 64$ for SHA-512.

The idea of feed-forward is taken one step further. We suggest the idea of providing feed-forward across message blocks. The intuitive justification is that this provides an additional mechanism for allowing the processing of the current block to depend on earlier blocks. Concrete suggestions are given for the SHA-2 family. These improve the resistance of SHA-2 hash functions against generic multi-collision attacks introduced in [28]. At a general level, our idea of feed-forward across message blocks can be seen as a practical version of the wide-pipe design strategy suggested in [32].

We put together all the ideas and make a new proposal called SShash. A description is given in Section 5.6 along with comparison of timing results to SHA-2. The implementation has been made by modifying the implementation of SHA-2 available from [11]. Test vectors and the code for the 256-bit variant of SShash are given in Section A.4.

Relation to the SHA-3 Competition. The NIST of USA is currently running a competition to select a new hash standard called SHA-3 [20]. Around 60 candidates have been submitted to the SHA-3 competition. Our proposal SShash has not been submitted to the SHA-3 competition. We intended to submit it to the SHA-3 competition, but missed the deadline for submission to the same.

According to NIST documentation [20], "NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard". So, scientific interest in SHA-2 is still very much alive. Our proposal should be seen in this light. We have suggested some modifications to SHA-2 so as to resist the recent reduced-round attacks and also to achieve other desirable features. As is the case in all designs of symmetric primitives, confidence in a primitive grows with the failure of cryptanalytic efforts. The same is also true for our proposal. If SShash gets broken in the future, then the procedure may throw some light on possible attacks on SHA-2. On the other hand, if the design survives, then some lessons would have been learnt.

The new proposal is not a competitor for SHA-3 and hence will not be a standard. But, NIST standards are of great interest worldwide. Consequently, there are users who would be interested in knowing how to avoid the recent reduced-round attacks on SHA-2. If SShash can indeed achieve this, then such users may consider using SShash for possible proprietary purposes.

5.2 Some Insights into Recent Attacks on SHA-2

Local collisions for linearized version of SHA-2 were studied by Gilbert and Handschuh [22] and in Chapter 3 of this thesis. All these local collisions hold for the actual SHA-256 with probabilities of about 2^{-39} or less. Using these local collisions, Mendel et al. [34,35] and later our work in Chapter 4 reported collisions for 18 round SHA-256. We call attacks using local collisions for the linearized version of SHA-2 as *linear attacks*.

Nikolić and Biryukov [38] presented a local collision which is valid for the actual SHA-256 function. The important point to note is that this local collision holds with probability of about 1/3. Using similar methods, we presented another local collision in Chapter 4 of this thesis which holds with probability 1. Extension of these attacks to obtain up to 24-round collisions for both SHA-256 and SHA-512 have been given in [26] and in Chapter 4 of this thesis. We call attacks using local collisions valid for the actual SHA-2 as *nonlinear attacks*.

We now discuss certain features of the SHA-2 design which facilitated collision attacks against reduced round versions discussed earlier.

Linear vs. Nonlinear attacks. The only places where XOR is used in the SHA-2 design are in the design of the transformations Σ_0, Σ_1 and σ_0, σ_1. Modular addition is much more extensively used in the round function. To a certain extent, this explains the difference in probabilities between linear and non-linear attacks.

Choice of the Transformations Σ_0 and Σ_1. Two transformations Σ_0 and Σ_1 are used in the round function of SHA-2. These transformations are given in Section 1.4.

Consider the equations $\Sigma_0(x) = x$ and $\Sigma_1(x) = x$. Any solution to these equations will give a "fixed point" for the transformations Σ_0 and Σ_1. Since both these transformations use only XORs, we can equivalently look at the equations $(\Sigma_0 \oplus I_{32})(x) = 0$ and $(\Sigma_1 \oplus I_{32})(x) = 0$ for SHA-256, where I_{32} is the identity matrix of order 32. For SHA-512, the I_{32} needs to be replaced by I_{64}.

For SHA-256, $\Sigma_0 \oplus I_{32}$ has rank 31 but $\Sigma_1 \oplus I_{32}$ has rank 29. The null space of $\Sigma_0 \oplus I_{32}$ has basis {`0xffffffff`}, whereas the null space of $\Sigma_1 \oplus I_{32}$ has basis {`0x99999999`, `0xaaaaaaaa`, `0xcccccccc`}. For SHA-512, the ranks of both $\Sigma_0 \oplus I_{64}$ and $\Sigma_1 \oplus I_{64}$ are 63 and the null space has basis {`0xffffffffffffffff`}.

Fixed points of Σ_0 and Σ_1 for both SHA-256 and SHA-512 are shown in Table 5.1.

Table 5.1: Fixed points of Σ_0 and Σ_1 for SHA-256 and SHA-512.

Hash function	Transformation	Fixed Points
SHA-256	Σ_0	{0x00000000, 0xffffffff}
	Σ_1	{0x00000000, 0xffffffff,
		0x33333333, 0x55555555,
		0x66666666, 0x99999999,
		0xaaaaaaaa, 0xcccccccc}
SHA-512	Σ_0	{0x0000000000000000,
		0xffffffffffffffff}
	Σ_1	{0x0000000000000000,
		0xffffffffffffffff}

The analysis above shows that both Σ_0 and Σ_1 have common fixed points for both the functions in the SHA-2 family. Moreover, the common fixed points have very simple structure as well: all the bits are either zero or one when they are expressed as 32-bit (or 64-bit) quantities. The numeric value of these common fixed points are 0 and -1. *This is one of the crucial issues which provides the nonlinear local collisions of high probability utilized in recent attacks.*

Cross Dependence Equation. The Cross Dependence Equation (CDE) was described in 4.1.1. We revisit the equation here. The CDE describes relationship between the e and the a register values in the update function of SHA-2 family as follows.

$$e_i \;=\; a_{i-4} + a_i - \Sigma_0(a_{i-1}) - f_{MAJ}(a_{i-1}, a_{i-2}, a_{i-3}).$$

The CDE allows an attacker to get simple relations between a and e registers by ensuring suitable behavior of f_{MAJ}. Note that it is rather easy to control the differential behavior of f_{MAJ} as utilized in [38] and other related works. The CDE, therefore, reduces the utility of two register updates in each round.

Local Collision and Message Expansion. The idea of perturbation-correction from [5] is used to obtain a local collision. If a message difference (perturbation) is introduced at Round i, then it is possible to define subsequent message differences such that the perturbation is cancelled at Round $i+8$. This leads to a 9-round local collision. The idea of the Nikolić-Biryukov attack [38] and its extensions for obtaining an r-round collision is the following. Choose a suitable i and place a local collision from Round i to $i+8$. Then ensure that $\delta W_j = 0$ for $j = i+9, \ldots, r-1$ leading to an

r-round collision. This approach succeeds because a perturbation introduced at some step can be "quickly" cancelled within a few steps. Viewed another way, the introduced perturbation need not affect registers at Round j if j is somewhat far from i, i.e., the perturbation does not have long-range effects.

5.3 Suggestions for Improvements to SHA-2 Design

5.3.1 Using Affine Transformations in the Update Function

We suggest that the linear functions Σ_0, Σ_1 be replaced by affine functions Γ_0 and Γ_1 respectively such that the following conditions are satisfied.

1. For all $x \in \{0,1\}^t$, $\Gamma_i(x) \neq x$ or $\overline{x}$, $i = 0, 1$.
2. For all $x \in \{0,1\}^t$, $\Gamma_0(x) \neq \Gamma_1(x)$ or $\overline{\Gamma_1(x)}$.

The first property is similar to one of the design criteria for the AES S-box [10]. This property ensures that Γ_i have no fixed points or complementary fixed point. On the other hand, the second property ensures that Γ_0 and Γ_1 do not agree upon any input; and also the output of one on any input cannot be obtained by complementing the output of the other on the same input. The formulation of these properties are motivated by the desire to avoid the situation where 0 and -1 are fixed points of both Σ_0 and Σ_1.

Achieving the above properties requires a bit of linear algebra. Suppose, we define $\Gamma_i(x) = \Sigma_i(x) \oplus \mathbf{b}_i$, where $\mathbf{b}_0, \mathbf{b}_1$ are to be chosen such that the above two conditions are satisfied. Additionally, we would like each of $\mathbf{b}_0$, $\mathbf{b}_1$ and $\mathbf{b}_0 \oplus \mathbf{b}_1$ to be either balanced or nearly balanced.

Consider the first point for SHA-256: $\Gamma_i(x) = x$ for some x implies $(\Sigma_i \oplus I_{32})x = \mathbf{b}_i$ and $\Gamma_i(x) = \overline{x}$ for some x implies $\Gamma_i(x) = (\Sigma_i \oplus I_{32})x = \overline{\mathbf{b}_i}$. In other words, the first point holds if both $\mathbf{b}_i$ and $\overline{\mathbf{b}_i}$ are not in the column space of $(\Sigma_i \oplus I_{32})$. In a similar manner, it can be shown that the second point holds if both $\mathbf{b}_0 \oplus \mathbf{b}_1$ and $\overline{\mathbf{b}_0 \oplus \mathbf{b}_1}$ are not in the column space of $(\Sigma_0 \oplus \Sigma_1)$.

We computed many choices of $\mathbf{b}_0$ and $\mathbf{b}_1$ satisfying these constraints. Examples for SHA-256 are

$$\mathbf{b}_0 = \texttt{0xdcb2344c} \text{ and } \mathbf{b}_1 = \texttt{0x9b097671}.$$

For SHA-512, examples are

$$\mathbf{b}_0 = \texttt{0x1762e66a04d6be32} \text{ and } \mathbf{b}_1 = \texttt{0x12135c7549e2fcdd}.$$

5.3.2 Mix of + and $\oplus$

In the design of the SHA-2 round function, the XOR operation is used a relatively lesser number of times compared to modular addition. A better mix of + and $\oplus$ can be obtained by using + to add W_i to obtain a_i and using $\oplus$ to XOR W_i to obtain e_i. This will mean that if we work with XOR differentials, then it will be difficult to analyze the XOR differentials of the a-register; on the other hand, if we work with modular differentials, then it will be difficult to analyze the modular differentials of the e-register.

A property of the SHA-2 round function is that it is easy to fix $a_{i-4}, \ldots, a_i$ (using words $W_{i-4}, \ldots, W_i$) to simple values and ensure that e_i is also fixed to a simple value. This is because of the CDE (4.1.1). An example of the simplification that is possible using the CDE is when $a_{i-1} = a_{i-2} = 0$. In this case, $e_i = a_i + a_{i-4}$, which is a rather simple relation.

To remove the above issues, we suggest the update functions to be the following.

$$\left.\begin{array}{rcl} a_i &=& h_{i-1} \oplus (\Gamma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Gamma_1(e_{i-1}) \\ & & +f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + K_i + W_i) \\ b_i &=& a_{i-1} \\ c_i &=& b_{i-1} \\ d_i &=& c_{i-1} \\ e_i &=& (d_{i-1} + \Gamma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i) \\ & & \oplus W_i \\ f_i &=& e_{i-1} \\ g_i &=& f_{i-1} \\ h_i &=& g_{i-1}. \end{array}\right\} \quad (5.3.1)$$

Note that the term $T_{i-1} = \Gamma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + K_i$ is present in the computation of both a_i and e_i. This common sub-expression need to be computed only once for each round. In the SHA-2 compression function, the term $\Sigma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i + W_i$ is common to both a_i and e_i. Computing Γ requires one extra t-bit XOR operation, where $t = 32$ for SHA-256 and $t = 64$ for SHA-512. By our estimate, the round function given in (5.3.1) requires 6 extra t-bit operations when compared to the SHA-2 round function. We expect this to have insignificant effect on the efficiency. The actual efficiency depends on a large number of parameters such as cache size, instruction pipelining, number of processor cores, etcetera.

New Cross-Dependence Relation. We have

$$\begin{array}{rcl} T_{i-1} &=& (a_i \oplus h_{i-1}) - (\Gamma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) - W_i) \\ &=& (e_i \oplus W_i) - (d_{i-1} + h_{i-1}). \end{array}$$

Consequently the new cross-dependence relation is the following.

$$\begin{aligned} e_i &= W_i \oplus ((a_i \oplus h_{i-1}) + (d_{i-1} + h_{i-1}) \\ &\quad -(\Gamma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) - W_i)) \end{aligned} \tag{5.3.2}$$

The dependences on both W_i and h_{i-1} do not directly cancel out. As a result, it is impossible to express e_i solely in terms of a_{i-4} to a_i. Going back to the previous example, if $a_{i-1} = a_{i-2} = 0$ (note $b_{i-1} = a_{i-2}$) then $e_i = W_i \oplus ((a_i \oplus h_{i-1}) + (d_{i-1} + h_{i-1} + W_i - \mathbf{b}_0))$ which has two alternations of $+$ and $\oplus$ and is a more complicated relation compared to the simple $e_i = a_i + a_{i-4}$ that is obtained for the SHA-2 compression function.

Resistance to Linear Attacks. Local collisions using linear approximations involve two steps. In the first step, all additions are replaced by XORs and in the second step, the functions f_{MAJ} and f_{IF} are replaced by suitable linear approximations [22, 47]. Then one considers the differential behavior of the resulting linear function and tries to obtain a local collision for the linearized version of the round function.

Let $\ell_1(x, y, z)$ and $\ell_2(x, y, z)$ be linear approximations of $f_{MAJ}(x, y, z)$ and $f_{IF}(x, y, z)$. Further, suppose all additions are replaced by XORs in the SHA-2 round function. Define $\Delta a_i = a'_i \oplus a_i$, where a'_i and a_i correspond to two different messages. Similarly, define Δ of the other registers. Then for the SHA-2 round function, we have,

$$\begin{aligned} \Delta a_i &= \Sigma_0(\Delta a_{i-1}) \oplus \ell_1(\Delta a_{i-1}, b_{i-1}, c_{i-1}) \oplus \Sigma_1(\Delta e_{i-1}) \\ &\quad \oplus \ell_2(\Delta e_{i-1}, f_{i-1}, g_{i-1}) \oplus \Delta h_{i-1} \oplus \Delta W_i; \end{aligned} \tag{5.3.3}$$

$$\begin{aligned} \Delta e_i &= \Delta d_{i-1} \oplus \Sigma_1(\Delta e_{i-1}) \oplus \ell_2(\Delta e_{i-1}, f_{i-1}, g_{i-1}) \\ &\quad \oplus \Delta h_{i-1} \oplus \Delta W_i. \end{aligned} \tag{5.3.4}$$

For the round function shown in (5.3.1), we have

$$\begin{aligned} \Gamma_0(a'_{i-1}) \oplus \Gamma_0(a_{i-1}) &= \Sigma_0(a'_{i-1}) \oplus \mathbf{b}_0 \oplus \Sigma_0(a_{i-1}) \oplus \mathbf{b}_0 \\ &= \Sigma_0(a'_{i-1}) \oplus \Sigma_0(a_{i-1}) \\ &= \Sigma_0(a'_{i-1} \oplus a_{i-1}) \\ &= \Sigma_0(\Delta a_{i-1}). \end{aligned}$$

A similar calculation shows that $\Gamma_1(e'_{i-1}) \oplus \Gamma_1(e_{i-1}) = \Sigma_1(\Delta e_{i-1})$. Now, it is easy to see that the expression for Δa_i and Δe_i for the round function shown in (5.3.1) are exactly those given by (5.3.3). As a result, the entire analysis of linear approximations [22, 47] done for SHA-2 compression function apply without any change to the new suggestion. In particular, this shows that the best linear local collision holds with probability 2^{-42} for the new suggestion as for the SHA-256 compression function.

Resistance to Non-Linear Attacks. In the SHA-2 round function, we have

$$\begin{aligned}\delta e_i &= \delta\Sigma_1(e_{i-1}) + \delta f_{IF}(\delta e_{i-1}, \delta f_{i-1}, \delta g_{i-1}) + \delta d_{i-1} + \delta h_{i-1} + \delta W_i \\ \delta a_i &= \delta\Sigma_0(a_{i-1}) + \delta f_{MAJ}(\delta a_{i-1}, \delta b_{i-1}, \delta c_{i-1}) + \delta e_i - \delta d_{i-1}.\end{aligned}$$

As a result, introducing a message difference of w at the i-th round, i.e., $\delta W_i = w$, makes $\delta a_i = \delta e_i = w$. This is the starting step of the Nikolić-Biryukov and later attacks. The next few message differences are used to cancel out the difference introduced in a_i and e_i. Let us now consider the differential form for a and e registers using the new suggestion.

$$\begin{aligned}\delta a_i &= a'_i - a_i \\ &= (h'_{i-1} \oplus (\Gamma_0(a'_{i-1}) + f_{MAJ}(a'_{i-1}, b'_{i-1}, c'_{i-1}) + \Gamma_1(e'_{i-1}) \\ &\quad + f_{IF}(e'_{i-1}, f'_{i-1}, g'_{i-1}) + K_i + W'_i)) \\ &\quad - (h_{i-1} \oplus (\Gamma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Gamma_1(e_{i-1}) \\ &\quad + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + K_i + W_i)) \\ \delta e_i &= e'_i - e_i \\ &= ((d'_{i-1} + \Gamma_1(e'_{i-1}) + f_{IF}(e'_{i-1}, f'_{i-1}, g'_{i-1}) + h'_{i-1} + K_i) \oplus W'_i) \\ &\quad - ((d_{i-1} + \Gamma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i) \oplus W_i).\end{aligned}$$

Suppose that we now introduce an additive difference of w in δW_i, i.e., $\delta W_i = W'_i - W_i = w$. Then

$$\begin{aligned}\delta a_i &= (c_1 \oplus (c_2 + W_i + w)) - (c_1 \oplus (c_2 + W_i)) \\ &\neq (c_3 \oplus (W_i + w)) - (c_3 \oplus W_i) = \delta e_i\end{aligned}$$

where

$$\begin{aligned}c_1 &= h_{i-1}, \\ c_2 &= \Gamma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Gamma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) \\ &\quad + K_i, \\ c_3 &= d_{i-1} + \Gamma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i.\end{aligned}$$

Since i is the first place where the perturbation is introduced, the quantities corresponding to primed and unprimed variables are same for $(i-1)$st round, i.e., the quantities c_1, c_2 and c_3 take the same value for both primed and unprimed values.

One can notice that for the new suggestion, the expressions for δa_i and δe_i are not as simple as that for SHA-2 compression functions. Introducing a perturbation of w through W_i does not introduce the same perturbation in the a and e registers and neither of these perturbations is equal to w. The analysis of a few more rounds does not provide any means to cancel out the perturbations of δa_i and δe_i. Consequently, there is no way to apply the Nikolić-Biryukov type attack on this round function. This is an improvement over the SHA-2 round function.

5.4 Multiple Feed Forward: A New Design Construct

We introduce the idea of multiple feed-forward, i.e., feed-forward at several places. This provides a possible additional layer of resistance against perturbation-correction attacks. The idea of using more than one feed-forward has also been used recently in the SHA-3 candidate Arirang [6]. We would like to note two points in this regard.

1. Though the basic idea of multiple feed-forwards is the same, the details of how the feed-forwards are applied and the feed-forward points are chosen are different.

2. Our idea of multiple feed-forwards is based on our earlier technical report [54]. At the time (June 2008) of posting of this technical report, the Arirang design was not publicly available. So, to the best of our understanding, the idea of using several feed-forwards has been obtained independently by us and by the designers of Arirang.

The computation starts with a C (the initial value of which is IV) and suppose that the output of round i is $S^{(i)}$. The SHA-2 compression function employs a single feed-forward, i.e., the output of H is $C+S^{(r-1)}$ where $r = 64$ for SHA-256 and $r = 80$ for SHA-512.

Let $t_1 = 7$, $t_2 = 14$ and $s = 16$. We introduce new feed-forwards at rounds $t_1 + s \times k_1$ and $t_2 + s \times k_2$, where k_1, k_2 are positive integers. Let $S^{(-1)} = C$. We use two sets of temporary registers T_1 and T_2, i.e., each T_i consists of $n = 8$ t-bit words, where $t = 32$ for SHA-256 and $t = 64$ for SHA-512.

The algorithm for the compression function is shown in Figure 5.2. In the figure assume for the moment that in the input $T_1 = T_2 = 0^{nt}$ and the output does not contain T_1 and T_2. The general description allows the introduction of feed-forward across message blocks as explained in the next section.

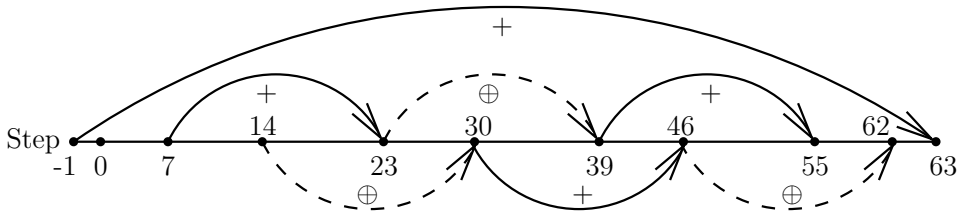
The following points are to be noted.

1. The feed-forward with C at Round $r - 1$ remains unchanged.

2. There are two feed-forward threads starting at t_1 and t_2. Both threads alternate the type of feed-forward between $+$ and $\oplus$. The first thread starts with $+$, while the second thread starts with $\oplus$.

3. The choice of t_1 and t_2 ensures that if we take any interval of length 9 in the range 0 to 15, then at least one of t_1 or t_2 will lie properly within the interval.

Figure 5.1: Illustration of multiple feed-forward for SHA-256. There are two threads of feed-forwards, based on XOR and modular addition. The original additive feed-forward from the input (marked as Round −1) to Round 63 is also retained.



4. The choices of t_1, t_2 and s ensure that the sets $\{t_1 + s \times k_1 : k_1 \geq 1\}$ and $\{t_2 + s \times k_2 : k_2 \geq 1\}$ are disjoint. (A sufficient condition for this is to have s to be co-prime to $(t_1 - t_2)$.) So, at no round will two feed-forward threads starting at t_1 and t_2 meet.

5. New feed-forward is not introduced at Round $(r-1)$. Again, the choices of t_1, t_2 and s ensure this.

6. The amortized cost of the new feed-forwards is less than one t-bit operation (addition or XOR) per round, where t is 32 for SHA-256 and 64 for SHA-512. In an efficient implementation with loop unrolling, the conditional statements will not be required. The temporary registers T_1 and T_2 will simply be added or XORed to the internal registers at the appropriate round.

For SHA-256, the line starting at $t_1 = 7$ introduces feed-forwards at Rounds 23, 39 and 55 and the line starting at $t_2 = 14$ introduces feed-forwards at Rounds 30, 46 and 62. Additive feed-forward is used at Rounds 23 and 55 due to the first line and at Round 46 due to the second line. XOR feed-forward is used at Round 39 due to the first line and at Rounds 30 and 62 due to the second line. There is a total of 6 feed-forward steps requiring a total of $6 \times 8 = 48$ 32-bit add/XOR operations. Amortized over the 64 rounds, the cost is 3/4th 32-bit operation per round. The feed-forwards for SHA-256 are illustrated in Figure 5.1.

For SHA-512, the feed-forwards of the first line occur at Rounds 23, 39, 55 and 71 with additive feed-forwards at Rounds 23 and 55 and XOR feed-forwards at Rounds 39 and 71. The second line of feed-forwards is introduced at Rounds 30, 46, 62 and 78, with XOR at Rounds 30 and 62 and addition at Rounds 46 and 78. Amortized over the 80 rounds, the cost is 4/5th 64-bit operation (add/XOR) per round.

The idea of multiple feed-forward is generic and can be used with other hash designs. Depending on the number of registers, the number of free

words and the total number of rounds r, the value of t_1, t_2 and s has to be chosen appropriately so as to ensure the properties described above.

Resistance to Perturbation-Correction Attacks. The basic idea of such an attack has been discussed earlier. By our choice of t_1 and t_2, at least one of these will be properly contained in any 9-round local collision. Consequently, the registers at the corresponding round will have a difference. Due to feed-forwards, this difference will be pushed out into the rounds where the message words are defined through message recursion. Since these words cannot be directly controlled, it will be difficult to cancel out the introduced perturbation at these rounds. The overall effect will be that the local collision based perturbation-correction attack will become substantially more difficult to apply.

A Possible Drawback. There is a possibility that the idea of multiple feed-forward can be used to attack the design. The feed-forward steps may be seen as providing possible control in the message expansion region which can in fact be used to correct perturbations in the higher rounds. However, we could not actually see how this could be done. At this point of time, we think multiple feed-forwards actually create perturbations in the message expansion region rather than cancel them.

5.5 Feed-Forward Across Message Blocks

The idea of feed-forward can be extended to different message blocks. Let us go back to the construction in Section 5.4. After the first message block has been processed, three quantities are produced, the chaining variable C and the quantities T_1 and T_2. But, T_1 and T_2 are not used further. These two quantities are "lightweight" digests of the first message block. We suggest that these be used in the processing of the second message block.

The processing of the second block starts with C as the IV and by the construction of Section 5.4, the output of Round 7 is taken to be new T_1 and the output of Round 14 is taken to be the new T_2. We modify this as follows. The new T_1 is the XOR of the old T_1 and the output of Round 7; and the new T_2 is the sum of the old T_2 and the output of Round 14. The rest of the two feed-forward threads are as before. Figure 5.2 shows the complete description. Note that at a general level, this idea of feed-forward across message blocks is similar to the wide-pipe design strategy introduced in [32].

The modification can still be considered to be within the MD framework. Let $M^{(0)}, \ldots, M^{(\ell)}$ be the message blocks (including padding with

Figure 5.2: Modified compression function with two feed-forward threads. Here $t_1 = 7$, $t_2 = 14$ and $s = 16$; G_i is the round function.

$\mathsf{Compress}(\mathrm{reg}, T_1, T_2, W)$
1. parse W into 16 t-bit words $W_0, \ldots, W_{15}$;
2. $(W_0, \ldots, W_{r-1}) = \mathsf{msgExpn}(W_0, \ldots, W_{15})$;
3. $S^{-1} = \mathrm{reg}$;
4. for $i = 0, \ldots, 15$ do
5. $\quad S^{(i)} \leftarrow G_i(S^{(i-1)}, W_i)$;
6. $\quad$ if $(i = 7)$ then $T_1 \leftarrow T_1 \oplus S^{(i)}$; if $(i = 14)$ then $T_2 \leftarrow T_2 + S^{(i)}$;
7. end for;
8. for $i = 16, \ldots, r-1$ do
9. $\quad S^{(i)} = G_i(S^{(i-1)}, W_i)$;
10. $\quad$ if $((i - t_1) \bmod s = 0)$ then $S^{(i)} \leftarrow S^{(i)} + T_1$; $T_1 = S^{(i)}$;
11. $\quad$ if $((i - t_1) \bmod 2s = 0)$ then $S^{(i)} \leftarrow S^{(i)} \oplus T_1$; $T_1 = S^{(i)}$;
12. $\quad$ if $((i - t_2) \bmod s = 0)$ then $S^{(i)} \leftarrow S^{(i)} \oplus T_2$; $T_2 = S^{(i)}$;
13. $\quad$ if $((i - t_2) \bmod 2s = 0)$ then $S^{(i)} \leftarrow S^{(i)} + T_2$; $T_2 = S^{(i)}$;
14. end for;
15. output $(S^{(r-1)} + \mathrm{reg}, T_1, T_2)$.

length). Let $C^{(-1)} = \mathsf{IV}$ and $T_1^{(-1)} = T_2^{(-1)} = 0^{nt}$. There are ℓ invocations of the compression function, where the $(i+1)$st invocation of the compression function takes $(C^{(i)}, T_1^{(i)}, T_2^{(i)}, M^{(i+1)})$ as input and produces a $(C^{(i+1)}, T_1^{(i+1)}, T_2^{(i+1)})$ as output. Here $C^{(i)}, C^{(i+1)}$ are the chaining variables; and $T_1^{(i)}, T_2^{(i)}, T_1^{(i+1)}, T_2^{(i+1)}$ are the feed-forward quantities.

Viewed in this way, it is easy to prove as usual by backward induction that a collision for the hash function leads to a collision for the compression function. Finding a pseudo collision for the compression function defined in this manner may be easier than the compression function where there is no feed-forward across message blocks. This is because one may choose T_1 and T_2 as suitable values. However, the hashing starts with $C^{(-1)} = \mathsf{IV}$ and $T_1^{(-1)} = T_2^{(-1)} = 0^{nt}$, which fixes the initial choice of T_1 and T_2.

Resistance to Multi-Collision Attacks. Let us now consider the effect of multi-collision attacks [28]. This is a generic technique which applies to the MD type construction. Using r invocations of generic collision finding algorithm on the compression function, it is possible to construct 2^r messages which map to the same value. Suppose M_1, M_1' are two distinct message blocks which (starting from IV) map to the same value C_1; and M_2, M_2' are two distinct message blocks which starting from C_1 map to the same value C_2. Then the four messages (M_1, M_2), (M_1, M_2'), (M_1', M_2) and (M_1', M_2')

map to the same value C_2. Since the output of the compression function consists of nt bits, the generic algorithm will require $2 \times 2^{nt/2}$ invocations of the compression function to find a collision.

Suppose we apply this to the new construction. The output of the compression function is the chaining variable C as well as T_1 and T_2. Then we need distinct M_1, M_1' which starting from IV and $T_1 = T_2 = 0^{nt}$ map to same value $C^{(1)}, T_1^{(1)}, T_2^{(1)}$; and distinct M_2, M_2' which starting from $C^{(1)}, T_1^{(1)}, T_2^{(1)}$ map to the same value $C^{(2)}, T_1^{(2)}, T_2^{(2)}$. Then as before, we will have four messages which map to the same value $C^{(2)}, T_1^{(2)}, T_2^{(2)}$. The advantage here is that the generic collision finding algorithm now needs to be applied to a compression function whose output is $3nt$ bits. As a result, the generic algorithm will require $2 \times 2^{3nt/2}$ invocations of the new compression function to find a collision. This is more than 2^n invocations and hence is not useful. (If a hash function produces n-bit digests, then one can *surely* obtain a collision by applying the hash function to $2^n + 1$ distinct inputs; and one can obtain a collision with *high probability* by applying the hash function to $2^{n/2}$ inputs.)

On the other hand, suppose that M_1, M_1' are such that starting from IV and $T_1 = T_2 = 0^{nt}$, they produce the same value for $C^{(1)}$ but not necessarily the same value for $T_1^{(1)}, T_2^{(1)}$, i.e., $(T_1^{(1)}, T_2^{(1)}) \neq ((T_1^{(1)})', (T_2^{(1)})')$. Further, suppose that M_2, M_2' are such that M_2 starting from $C^{(1)}$ and $T_1^{(1)}, T_2^{(1)}$ produces the same $C^{(2)}$ that M_2' starting from $C^{(1)}$ and $(T_1^{(1)})', (T_2^{(1)})'$ produces. This results in single two-block collision between (M_1, M_1') and (M_2, M_2'). But, now (M_1, M_2) does not produce the same value as (M_1, M_2'). More generally, no two of the four possible messages produce the same value. This is due to the difference in the intermediate feed-forward values, i.e., $(T_1^{(1)}, T_2^{(1)}) \neq ((T_1^{(1)})', (T_2^{(1)})')$. To summarize, in a manner similar to the wide-pipe design strategy [32], extending feed-forward across message blocks provides resistance to generic multi-collision attacks.

5.6 Design Specification and Implementation

We describe the complete hash algorithm, which we call SShash. There are two main variants – SShash-256 and SShash-512 with truncated versions SShash-224 and SShash-384 being obtained as in SHA-2 [59].

Suppose M is a message to be hashed. We consider M to be a binary string of length λ and this string is padded as in SHA-2, i.e., append the bit 1 to the message, followed by k zeros where k is the smallest non-negative integer such that $\lambda + 1 + k \equiv 14t \bmod 16t$, with $t = 32$ for SShash-256 and $t = 64$ for SShash-512. Then the $2t$-bit binary representation of λ is appended. This makes the total length of the padded message to be a multiple of $2t$. The padded message is parsed into q $16t$-bit blocks $M^{(1)}, \ldots, M^{(q)}$.

The algorithm starts with 8 t-bit registers called the initialization vector

IV; the IV for SShash-256 is same as the IV for SHA-256 and the IV for SShash-512 is same as the IV for SHA-512. A compression function Compress is used to process the message blocks and produce the digest in the following manner.

SShash.
1. reg = IV; $T_1 = T_2 = 0^{8t}$;
2. for $i = 1$ to q do $(\text{reg}, T_1, T_2) = \text{Compress}(\text{reg}, T_1, T_2, M^{(i)})$;
4. return reg;

The function Compress is shown in Figure 5.2. It takes as input 8 t-bit registers and a $16t$-bit message block. The message block is parsed into 16 t-bit words $W_0, \ldots W_{15}$. These words are then expanded into r t-bit words $W_0, \ldots, W_{r-1}$ using msgExpn. The message expansion for SShash-256 is the same as that for SHA-256 and the message expansion for SShash-512 is the same as that for SHA-512. In particular, the value of r is 64 for SShash-256 and 80 for SShash-512. (Note. In the above q refers to the number of message blocks, while r refers to the number of t-bit words after the expansion of a single message block.)

The round function G is applied r times. This function G takes as input 8 t-bit registers $(a_{i-1}, \ldots, h_{i-1})$ and updates them using W_i to obtain $(a_i, \ldots, h_i)$. This updation is done as shown in (5.3.1). These updations require the values of the t-bit constants $K_0, \ldots, K_{r-1}$. Again, these constant values are as specified in SHA-2.

The above completes the description of the modified algorithm SShash. We have implemented both SShash-256 and SShash-512. For this implementation, we modified the SHA-2 code available at [11]. Macros for the compression function of SShash-256 is given in Appendix A.4. The code for SShash-512 is similar and is not shown. Appendix A.4 also provides test vectors for SShash.

Speed comparison of SShash with SHA-2 is shown in Table 5.2. The obtained speeds correspond with our earlier stated intuition that the suggested modifications of SHA-2 to obtain SShash do not affect the efficiency too much.

5.7 Conclusions

In this chapter, we studied several properties of the SHA-2 design which were used in recent collision attacks against reduced SHA-2. We have suggested modifications to the SHA-2 design so as to make these attacks inapplicable. The modified SHA-2 design is almost as efficient as the original one.

We provided a generic construction of multiple feed-forward. This can be used to strengthen a design against perturbation-correction collision finding

Table 5.2: Speed of SShash compared to SHA-2. For SHA-2, the implementation from [11] has been used and SShash implementations have been obtained by modifying the SHA-2 implementations of [11]. The speed measurements are given in cycles/byte and were obtained on a dual-core Pentium having two CPUs at 2.2 GHz and running Fedora.

SHA-256	SShash-256	SHA-512	SShash-512
19.3	22	13.6	14.0

attacks. Further, the idea of feed-forward over several message blocks is suggested and shown to provide resistance to multi-collision attacks.

Chapter 6

Conclusions and Open Problems

In this thesis we have made a systematic study of linear as well as non-linear local collisions for the SHA-2 family. We utilized these different types of local collisions to obtain reduced round attacks on the members of the SHA-2 family. Using linearized local collisions, we presented an algorithm for finding 18-round SHA-256 collisions. Later, using the techniques of Nikolić and Biryukov [38], we studied non-linear local collisions for SHA-2 family. We presented a unified combinatorial analysis of 9-round nonlinear local collisions for SHA-2 family. Utilizing a nonlinear local collision, we obtained deterministic attacks against up to 22-round SHA-2 family. Finally we extended these attacks to 23 and 24 rounds of SHA-256 and SHA-512. Our 23 and 24-round SHA-2 attacks have a simpler description than the earlier known 23 and 24-round attacks described in [26]. These also improve on the complexities of the earlier corresponding round attacks. All our attacks utilizing the nonlinear local collision were studied under a unified model of shifting the local collision one step at a time. This is unlike the attacks of [26] where a pseudo collision is first constructed, which is later converted into a collision for appropriate number of rounds.

Building on the work of finding collisions for reduced round SHA-2 family, we suggested improvements to the SHA-2 family to make them immune to all the known attacks. Our suggestions do not cause much performance degradation but they are likely to enhance the security of the SHA-2 design. One of our suggestions, namely the Multiple Feed Forward, is quite generic and can be used with other hash designs as well.

We hope this work will help understand the SHA-2 family better and help construct longer round attacks against it. We also hope it will help in designing new and better hash functions for the future.

There are several ways in which the work described in this thesis can be extended. Some of the open problems on the area are listed below.

1. We have seen that a single local collision cannot yield a collision better than 24 rounds of SHA-2. To go farther will require a combination of multiple local collisions, and possibly the combination of linear and non-linear techniques. How to achieve this is not yet known.

2. We have not attempted pre-image or second pre-image attacks in this thesis. Investigation and development of such attacks against SHA-2 will be an interesting problem.

3. We have also not tried to look for other non-random properties of SHA-2 design, like pseudo collisions or near collisions. Exhibiting non-randomness of the SHA-2 design could also be a possible extension of this work.

Bibliography

[1] Ross J. Anderson and Eli Biham. TIGER: A Fast New Hash Function. In Gollmann [23], pages 89–97.

[2] Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Franklin [21], pages 290–305.

[3] Gilles Brassard, editor. *Advances in Cryptology - CRYPTO 1989, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.

[4] Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[5] Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO 1998, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998.

[6] D. Chang, S. Hong, C. Kang, J. Kang, J. Kim, C. Lee, J. Lee, J. Lee, S. Lee, Y. Lee, J. Lim, and J. Sung. ARIRANG: SHA-3 Proposal. *NIST SHA-3 candidate*, 2009. Available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/ARIRANG.zip`.

[7] RIPE Consortium. Ripe Integrity Primitives. *Final report of RACE Integrity Primitives Evaluation (R1040)*, 1007, 1995.

[8] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[9] Joan Daemen and Craig S. K. Clapp. Fast Hashing and Stream Encryption with PANAMA. In Vaudenay [63], pages 60–74.

[10] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[11] Wei Dai. Crypto++ Library 5.5.2. `http://www.cryptopp.com/`.

[12] Ivan Damgård. A Design Principle for Hash Functions. In Brassard [3], pages 416–427.

[13] Alain Daniélou. *The Complete Kamasutra (Translation)- Original Sanskrit Text by Vatsyayana*. Inner Traditions, 1993.

[14] Magnus Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-Universitat Bochum, 2005. Available at `http://www.cits.rub.de/imperia/md/content/magnus/dissmd4.pdf`.

[15] Bert den Boer and Antoon Bosselaers. An Attack on the Last Two Rounds of MD4. In Joan Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 1991.

[16] Whitefield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[17] Hans Dobbertin. Cryptanalysis of MD4. *Journal of Cryptology*, 11(4):253–271, 1998.

[18] Hans Dobbertin. The First Two Rounds of MD4 are Not One-Way. In Vaudenay [63], pages 284–292.

[19] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In Gollmann [23], pages 71–82.

[20] Federal Register Vol. 72, No. 212. *Announcing Request for Candidate Algorithm Nominations for a new Cryptographic Hash Algorithm (SHA-3) Family*. U.S. Department of Commerce, National Institute of Standards and Technology(NIST), November 2, 2007. Available at `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`. Details of the SHA-3 competition are available at `www.nist.gov/hash-competition`.

[21] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.

[22] Henri Gilbert and Helena Handschuh. Security Analysis of SHA-256 and Sisters. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 175–193. Springer, 2003.

[23] Dieter Gollmann, editor. *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*. Springer, 1996.

[24] Philip Hawkes, Michael Paddon, and Gregory G. Rose. On Corrective Patterns for the SHA-2 Family. *Cryptology eprint Archive*, August 2004. Available at `http://eprint.iacr.org/2004/207`.

[25] Marko Hölbl, Christian Rechberger, and Tatjana Welzer. Finding Message Pairs Conforming to Simple SHA-256 Characteristics. In Stefan Lucks, Ahmad-Reza Sadeghi, and Christopher Wolf, editors, *Preliminary Proceeding Records of WEWoRC 2007 - Western European Workshop on Research in Cryptology, July 4-6, 2007, Bochum, Germany*, pages 21–25, 2007.

[26] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other Non-Random Properties for Step-Reduced SHA-256. In *Selected Areas in Cryptography, 15th Annual International Workshop, SAC 2008, Revised Papers*, 2008. To appear.

[27] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other Non-Random Properties for Step-Reduced SHA-256. *Cryptology eprint Archive*, April 2008. Available at `http://eprint.iacr.org/2008/131`.

[28] Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Franklin [21], pages 306–316.

[29] David Kahn. *The Codebreakers: The Story of Secret Writing*. Scribner, 1996.

[30] Donald Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley Professional, 2nd edition, 1997.

[31] Jeffrey S. Leon. A Probabilistic Algorithm for Computing Minimum Weights of Large Error-Correcting Codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

[32] Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer, 2005.

[33] Krystian Matusiewicz, Josef Pieprzyk, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of simplified variants of SHA-256. In Christopher Wolf, Stefan Lucks, and Po-Wah Yau, editors, *WEWoRC 2005 - Western European Workshop on Research in Cryptology, July 5-7, 2005, Leuven, Belgium*, volume 74 of *LNI*, pages 123–134. GI, 2005.

[34] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of Step-Reduced SHA-256. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2006.

[35] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of Step-Reduced SHA-256. *Cryptology eprint Archive*, March 2008. Available at `http://eprint.iacr.org/2008/130`.

[36] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. Available at `http://www.cacr.math.waterloo.ca/hac/`.

[37] Ralph C. Merkle. One Way Hash Functions and DES. In Brassard [3], pages 428–446.

[38] Ivica Nikolić and Alex Biryukov. Collisions for Step-Reduced SHA-256. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, March 26-28, 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2008.

[39] Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 78–95. Springer, 2005.

[40] Vincent Rijmen and Paulo Barreto. Whirlpool Hash Function. *International Organization for Standardization, ISO/IEC Standard 10118-3*, 2004.

[41] Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 58–71. Springer, 2005.

[42] Ronald L. Rivest. The MD4 Message Digest Algorithm. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer, 1990.

[43] Ronald L. Rivest. The MD4 message-digest algorithm, Request for Comments (RFC) 1320. *Internet Activities Board*, 1992.

[44] Ronald L. Rivest. The MD5 message-digest algorithm, Request for Comments (RFC) 1321. *Internet Activities Board*, 1992.

[45] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[46] Somitra Kumar Sanadhya and Palash Sarkar. 22-step Collisions for SHA-2. arXiv e-print archive, arXiv:0803.1220v1, March 2008. Available at `http://de.arxiv.org/abs/0803.1220`.

[47] Somitra Kumar Sanadhya and Palash Sarkar. New Local Collisions for the SHA-2 Hash Family. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings*, volume 4817 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2007.

[48] Somitra Kumar Sanadhya and Palash Sarkar. Attacking Reduced Round SHA-256. In Steven Bellovin and Rosario Gennaro, editors, *Applied Cryptography and Network Security - ACNS 2008, 6th International Conference, New York, NY, June 03-06, 2008, Proceedings*, volume 5037 of *Lecture Notes in Computer Science*, pages 130–143. Springer, 2008.

[49] Somitra Kumar Sanadhya and Palash Sarkar. Deterministic Constructions of 21-Step Collisions for the SHA-2 Hash Family. In Editors, editor, *Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 2008, Proceedings*, volume 5222 of *Lecture Notes in Computer Science*, pages 244–259. Springer, 2008.

[50] Somitra Kumar Sanadhya and Palash Sarkar. New Collision Attacks Against Up To 24-step SHA-2 . In D.R. Chowdhury, V. Rijmen, and

A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India*, volume 5365 of *Lecture Notes in Computer Science*, pages 91–103. Springer, 2008.

[51] Somitra Kumar Sanadhya and Palash Sarkar. Non-Linear Reduced Round Attacks Against SHA-2 Hash family. In Yi Mu and Willy Susilo, editors, *Information Security and Privacy - ACISP 2008, The 13th Australasian Conference, Wollongong, Australia, 7-9 July 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2008.

[52] Somitra Kumar Sanadhya and Palash Sarkar. A Combinatorial Analysis of Recent Attacks on Step Reduced SHA-2 Family. *Cryptography and Communications - Discrete Structures, Boolean Functions and Sequences*, Accepted for publication, 2009.

[53] Somitra Kumar Sanadhya and Palash Sarkar. A New Hash Family Obtained by Modifying the SHA-2 Family. In Rei Safavi-Naini and Vijay Varadharajan, editors, *ASIACCS 2009, 4th International Conference, March 10-12, 2009, Sydney, Australia*, volume To appear. ACM Press, 2009.

[54] Somitra Kumar Sanadhya and Palash Sarkar. Some Observations on Strengthening the SHA-2 Family. *Cryptology eprint Archive*, June 2008. Available at `http://eprint.iacr.org/2008`.

[55] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[56] Victor Shoup, editor. *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*. Springer, 2005.

[57] Secure Hash Standard. *Federal Information Processing Standard Publication 180*. U.S. Department of Commerce, National Institute of Standards and Technology(NIST), May 1993.

[58] Secure Hash Standard. *Federal Information Processing Standard Publication 180-1*. U.S. Department of Commerce, National Institute of Standards and Technology(NIST), April 1994. Both [58] and [59] are now jointly available as FIPS 180-3 at : `http://www.csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf`.

[59] Secure Hash Standard. *Federal Information Processing Standard Publication 180-2*. U.S. Department of Commerce, National

Institute of Standards and Technology(NIST), 2002. Available at `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`.

[60] Jacques Stern. A Method for Finding Codewords of Small Weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

[61] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2nd edition, 1997.

[62] Serge Vaudenay. On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In Bart Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 1994.

[63] Serge Vaudenay, editor. *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*. Springer, 1998.

[64] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer [8], pages 1–18.

[65] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Shoup [56], pages 17–36.

[66] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Cramer [8], pages 19–35.

[67] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Shoup [56], pages 1–16.

[68] Stephen Wolfram. *The Mathematica Book*. Wolfram Media, 5th edition, 2003. `http://www.wolfram.com`.

[69] Hirotaka Yoshida and Alex Biryukov. Analysis of a SHA-256 Variant. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, volume 3897 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2005.

[70] Hongbo Yu. Non-randomness of 39-step SHA-256. Talk presented at the Rump Session of Eurocrypt 2008, April 2008.

[71] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. HAVAL - A One-Way Hashing Algorithm with Variable Length of Output. In Jennifer Seberry and Yuliang Zheng, editors, *ASIACRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104. Springer, 1992.

Appendix A

A.1 Differential Paths for the Linearized Local Collisions

Table A.1: Differential paths for the cases of Table 2.4. Probability calculations are done taking x to be 2^{31} for SHA-256 and 2^{63} for SHA-512.

Step	Registers								Case 5		Case 6		Case 7		Case 8	
i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr
1	x	0	0	0	x	0	0	0	x	1	x	1	x	1	x	1
2	0	x	0	0	$\Sigma_0(x)$	x	0	0	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$
3	0	0	x	0	x	$\Sigma_0(x)$	x	0	$\Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$x \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$\Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$
4	0	0	0	x	0	x	$\Sigma_0(x)$	x	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_1(x)$	$\frac{1}{2^8}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_1(x)$	$\frac{1}{2^8}$
5	0	0	0	0	x	0	x	$\Sigma_0(x)$	0	$\frac{1}{2^4}$	$x \oplus \Sigma_0(x)$	$\frac{1}{2^7}$	0	$\frac{1}{2^4}$	$x \oplus \Sigma_0(x)$	$\frac{1}{2^7}$
6	0	0	0	0	0	x	0	x	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$
7	0	0	0	0	0	0	x	0	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$
8	0	0	0	0	0	0	0	x	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$
9	0	0	0	0	0	0	0	0	x	1	x	1	x	1	x	1
Total Probability										$\frac{1}{2^{49}}$		$\frac{1}{2^{49}}$		$\frac{1}{2^{52}}$		$\frac{1}{2^{52}}$

Table A.2: Differential paths for the cases of Table 2.4. Probability calculations are done taking x to be 2^{31} for SHA-256 and 2^{63} for SHA-512.

Step	Registers								Case 9		Case 10		Case 11		Case 12	
i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr
1	x	0	0	0	x	0	0	0	x	1	x	1	x	1	x	1
2	0	x	0	0	$\Sigma_0(x)$	x	0	0	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$
3	0	0	x	0	0	$\Sigma_0(x)$	x	0	$x \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$\Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{14}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$\Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$
4	0	0	0	x	x	0	$\Sigma_0(x)$	x	$x \oplus \Sigma_0(x)$	$\frac{1}{2^8}$	0	$\frac{1}{2^5}$	$x \oplus \Sigma_0(x)$	$\frac{1}{2^8}$	0	$\frac{1}{2^5}$
5	0	0	0	0	x	x	0	$\Sigma_0(x)$	$x \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{10}}$	$\Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{10}}$
6	0	0	0	0	0	x	x	0	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^7}$
7	0	0	0	0	0	0	x	x	x	1	x	1	x	1	x	1
8	0	0	0	0	0	0	0	x	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$
9	0	0	0	0	0	0	0	0	x	1	x	1	x	1	x	1
Total Probability										$\frac{1}{2^{48}}$		$\frac{1}{2^{48}}$		$\frac{1}{2^{51}}$		$\frac{1}{2^{51}}$

Table A.3: Differential paths for the cases of Table 2.4. Probability calculations are done taking x to be 2^{31} for SHA-256 and 2^{63} for SHA-512.

Step	Registers								Case 13		Case 14		Case 15		Case 16	
i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr	ΔW_i	Pr
1	x	0	0	0	x	0	0	0	x	1	x	1	x	1	x	1
2	0	x	0	0	$x \oplus \Sigma_0(x)$	x	0	0	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$
3	0	0	x	0	x	$x \oplus \Sigma_0(x)$	x	0	$\Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$x \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{17}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{20}}$	$\Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x))$	$\frac{1}{2^{20}}$
4	0	0	0	x	x	x	$x \oplus \Sigma_0(x)$	x	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$\Sigma_1(x)$	$\frac{1}{2^{8}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{11}}$	$x \oplus \Sigma_1(x)$	$\frac{1}{2^{8}}$
5	0	0	0	0	x	x	x	$x \oplus \Sigma_0(x)$	$\Sigma_1(x)$	$\frac{1}{2^{7}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{10}}$	$x \oplus \Sigma_1(x)$	$\frac{1}{2^{7}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{10}}$
6	0	0	0	0	0	x	x	x	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{7}}$	$\Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{7}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{7}}$	$x \oplus \Sigma_0(x) \oplus \Sigma_1(x)$	$\frac{1}{2^{7}}$
7	0	0	0	0	0	0	x	x	0	1	0	1	0	1	0	1
8	0	0	0	0	0	0	0	x	x	$\frac{1}{2}$	0	$\frac{1}{2}$	x	$\frac{1}{2}$	0	$\frac{1}{2}$
9	0	0	0	0	0	0	0	0	x	1	x	1	x	1	x	1
Total Probability										$\frac{1}{2^{54}}$		$\frac{1}{2^{54}}$		$\frac{1}{2^{57}}$		$\frac{1}{2^{57}}$

Table A.4: 20 step linear DP for SHA-256. There are 31 SS_5 local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	00210808	00210808	0	0	0	00210808	0	0	0
1	f4ec270b	0100c000	00210808	0	0	c568a30a	00210808	0	0
2	fd5cb0fc	00000204	0100c000	00210808	0	0361320e	c568a30a	00210808	0
3	0f1c9c64	12080000	00000204	0100c000	00210808	0320d081	0361320e	c568a30a	00210808
4	e38e0ddb	40080208	12080000	00000204	0100c000	64ab980c	0320d081	0361320e	c568a30a
5	91de6968	00040600	40080208	12080000	00000204	3344e7c2	64ab980c	0320d081	0361320e
6	552353b0	00006000	00040600	40080208	12080000	601161ac	3344e7c2	64ab980c	0320d081
7	9b7f2ad2	26239208	00006000	00040600	40080208	35af8c0b	601161ac	3344e7c2	64ab980c
8	694d62fd	0	26239208	00006000	00040600	5789970e	35af8c0b	601161ac	3344e7c2
9	3c97a984	0	0	26239208	00006000	26279408	5789970e	35af8c0b	601161ac
10	85cea813	0	0	0	26239208	00006000	26279408	5789970e	35af8c0b
11	13b8198f	0	0	0	0	26239208	00006000	26279408	5789970e
12	27dcb927	0	0	0	0	0	26239208	00006000	26279408
13	00040600	0	0	0	0	0	0	26239208	00006000
14	00006000	0	0	0	0	0	0	0	26239208
15	26239208	0	0	0	0	0	0	0	0
16-19	0	0	0	0	0	0	0	0	0

A.2 Differential Paths Obtained Using Coding -Theoretic Methods

A.3 Colliding Message Pairs

Table A.5: 20 step linear DP for SHA-256. There are 33 GH local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	11000200	11000200	0	0	0	11000200	0	0	0
1	921ea8c4	46110000	11000200	0	0	525988c4	11000200	0	0
2	71813ea9	01000002	46110000	11000200	0	54867192	525988c4	11000200	0
3	d8d10826	60a14800	01000002	46110000	11000200	e0f14804	54867192	525988c4	11000200
4	b9f28f71	1e420280	60a14800	01000002	46110000	d2495608	e0f14804	54867192	525988c4
5	8f9b9016	00a00200	1e420280	60a14800	01000002	5d2b70c9	d2495608	e0f14804	54867192
6	6d415897	00004084	00a00200	1e420280	60a14800	91204504	5d2b70c9	d2495608	e0f14804
7	c7276fd3	000000a0	00004084	00a00200	1e420280	65834883	91204504	5d2b70c9	d2495608
8	0b152ad9	0	000000a0	00004084	00a00200	1b4082a8	65834883	91204504	5d2b70c9
9	08044ede	0	0	000000a0	00004084	00a00200	1b4082a8	65834883	91204504
10	8123d10c	0	0	0	000000a0	00004084	00a00200	1b4082a8	65834883
11	65230b89	0	0	0	0	000000a0	00004084	00a00200	1b4082a8
12	8f40d2aa	0	0	0	0	0	000000a0	00004084	00a00200
13	00a00200	0	0	0	0	0	0	000000a0	00004084
14	00004084	0	0	0	0	0	0	0	000000a0
15	000000a0	0	0	0	0	0	0	0	0
16-19	0	0	0	0	0	0	0	0	0

Table A.6: 21 step linear DP for SHA-256. There are 45 SS_5 local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	0100000a	0100000a	0	0	0	0100000a	0	0	0
1	23510596	0a050090	0100000a	0	0	8a152096	0100000a	0	0
2	926e15c7	c2041880	0a050090	0100000a	0	d10748ae	8a152096	0100000a	0
3	6b6f5fa5	22a00008	c2041880	0a050090	0100000a	cc401590	d10748ae	8a152096	0100000a
4	4bcb749e	00035100	22a00008	c2041880	0a050090	4bee7c02	cc401590	d10748ae	8a152096
5	1b692042	8480040c	00035100	22a00008	c2041880	2961d0ce	4bee7c02	cc401590	d10748ae
6	2c749ed0	26422000	8480040c	00035100	22a00008	e5117e91	2961d0ce	4bee7c02	cc401590
7	78a6949f	05014062	26422000	8480040c	00035100	a230feee	e5117e91	2961d0ce	4bee7c02
8	70cf2020	0	05014062	26422000	8480040c	a1108106	a230feee	e5117e91	2961d0ce
9	3c408d06	0	0	05014062	26422000	8181446e	a1108106	a230feee	e5117e91
10	b375fdee	0	0	0	05014062	26422000	8181446e	a1108106	a230feee
11	023c7a57	0	0	0	0	05014062	26422000	8181446e	a1108106
12	83a6352d	0	0	0	0	0	05014062	26422000	8181446e
13	8480040c	0	0	0	0	0	0	05014062	26422000
14	26422000	0	0	0	0	0	0	0	05014062
15	05014062	0	0	0	0	0	0	0	0
16-20	0	0	0	0	0	0	0	0	0

Table A.7: 21 step linear DP for SHA-256. There are 45 GH local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	0002009c	0002009c	0	0	0	0002009c	0	0	0
1	ee6bb675	80090000	0002009c	0	0	8cebf037	0002009c	0	0
2	a91c7e24	00201514	80090000	0002009c	0	0426575c	8cebf037	0002009c	0
3	e2c68750	42816080	00201514	80090000	0002009c	6a7d34c5	0426575c	8cebf037	0002009c
4	ecac961f	4e100262	42816080	00201514	80090000	5f324fdf	6a7d34c5	0426575c	8cebf037
5	13c4cbce	40000200	4e100262	42816080	00201514	0096fb20	5f324fdf	6a7d34c5	0426575c
6	c742bbcf	6c113420	40000200	4e100262	42816080	6c3b20b4	0096fb20	5f324fdf	6a7d34c5
7	4a23a824	04240100	6c113420	40000200	4e100262	b872cdb1	6c3b20b4	0096fb20	5f324fdf
8	8f8f731c	0	04240100	6c113420	40000200	d71d2312	b872cdb1	6c3b20b4	0096fb20
9	a701e563	0	0	04240100	6c113420	40000200	d71d2312	b872cdb1	6c3b20b4
10	2d32209c	0	0	0	04240100	6c113420	40000200	d71d2312	b872cdb1
11	b5551b71	0	0	0	0	04240100	6c113420	40000200	d71d2312
12	e50db794	0	0	0	0	0	04240100	6c113420	40000200
13	40000200	0	0	0	0	0	0	04240100	6c113420
14	6c113420	0	0	0	0	0	0	0	04240100
15	04240100	0	0	0	0	0	0	0	0
16-20	0	0	0	0	0	0	0	0	0

Table A.8: 22 step linear DP for SHA-256. There are 60 SS_5 local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	20088104	20088104	0	0	0	20088104	0	0	0
1	7723d081	61404100	20088104	0	0	43677185	20088104	0	0
2	974e080e	10042040	61404100	20088104	0	215bba83	43677185	20088104	0
3	c03afd79	05a14180	10042040	61404100	20088104	726188f1	215bba83	43677185	20088104
4	301bc5d8	420111c8	05a14180	10042040	61404100	fa63cdf0	726188f1	215bba83	43677185
5	a13271fd	43008748	420111c8	05a14180	10042040	bd64f2ba	fa63cdf0	726188f1	215bba83
6	a3795c7f	1d044014	43008748	420111c8	05a14180	679e6946	bd64f2ba	fa63cdf0	726188f1
7	5d086433	398a1838	1d044014	43008748	420111c8	69ca76a3	679e6946	bd64f2ba	fa63cdf0
8	dbcb0f3a	0	398a1838	1d044014	43008748	b8c6fb64	69ca76a3	679e6946	bd64f2ba
9	702d2d4f	0	0	398a1838	1d044014	7a8a9f70	b8c6fb64	69ca76a3	679e6946
10	b5f25131	0	0	0	398a1838	1d044014	7a8a9f70	b8c6fb64	69ca76a3
11	c3975255	0	0	0	0	398a1838	1d044014	7a8a9f70	b8c6fb64
12	872fbe4f	0	0	0	0	0	398a1838	1d044014	7a8a9f70
13	43008748	0	0	0	0	0	0	398a1838	1d044014
14	1d044014	0	0	0	0	0	0	0	398a1838
15	398a1838	0	0	0	0	0	0	0	0
16-21	0	0	0	0	0	0	0	0	0

Table A.9: 22 step linear DP for SHA-256. There are 59 GH local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	40200502	40200502	0	0	0	40200502	0	0	0
1	91474f60	10034314	40200502	0	0	280d4a54	40200502	0	0
2	01ac1d4e	0c800345	10034314	40200502	0	1d2c03da	280d4a54	40200502	0
3	c8319061	14021803	0c800345	10034314	40200502	4d0768e0	1d2c03da	280d4a54	40200502
4	41c0f00e	12111224	14021803	0c800345	10034314	5f493d66	4d0768e0	1d2c03da	280d4a54
5	29db56e8	41122608	12111224	14021803	0c800345	80fd2155	5f493d66	4d0768e0	1d2c03da
6	f592f204	82031028	41122608	12111224	14021803	e61db37a	80fd2155	5f493d66	4d0768e0
7	caaa1ee6	a0340814	82031028	41122608	12111224	19b2660d	e61db37a	80fd2155	5f493d66
8	cb37951b	0	a0340814	82031028	41122608	aa994301	19b2660d	e61db37a	80fd2155
9	aac397a4	0	0	a0340814	82031028	41122608	aa994301	19b2660d	e61db37a
10	8f02dd86	0	0	0	a0340814	82031028	41122608	aa994301	19b2660d
11	bf223e6e	0	0	0	0	a0340814	82031028	41122608	aa994301
12	e0899ff0	0	0	0	0	0	a0340814	82031028	41122608
13	41122608	0	0	0	0	0	0	a0340814	82031028
14	82031028	0	0	0	0	0	0	0	a0340814
15	a0340814	0	0	0	0	0	0	0	0
16-21	0	0	0	0	0	0	0	0	0

Table A.10: 23 step linear DP for SHA-256. There are 75 SS_5 local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	80018c11	80018c11	0	0	0	80018c11	0	0	0
1	e08ba9dd	22c18214	80018c11	0	0	247da71c	80018c11	0	0
2	9acf831a	29105222	22c18214	80018c11	0	b708f831	247da71c	80018c11	0
3	99b76e84	81804506	29105222	22c18214	80018c11	f95c13bc	b708f831	247da71c	80018c11
4	ef3d10f6	5217b401	81804506	29105222	22c18214	72466d77	f95c13bc	b708f831	247da71c
5	f2ce8282	5421e110	5217b401	81804506	29105222	5d3f5ef7	72466d77	f95c13bc	b708f831
6	15e6cb63	84815301	5421e110	5217b401	81804506	65882d39	5d3f5ef7	72466d77	f95c13bc
7	ee97bfc1	64196841	84815301	5421e110	5217b401	4dd8ba8f	65882d39	5d3f5ef7	72466d77
8	6d60f25f	0	64196841	84815301	5421e110	a83a984b	4dd8ba8f	65882d39	5d3f5ef7
9	4e6744df	0	0	64196841	84815301	30388951	a83a984b	4dd8ba8f	65882d39
10	bf10f8de	0	0	0	64196841	84815301	30388951	a83a984b	4dd8ba8f
11	5b6b267a	0	0	0	0	64196841	84815301	30388951	a83a984b
12	2db30d74	0	0	0	0	0	64196841	84815301	30388951
13	5421e110	0	0	0	0	0	0	64196841	84815301
14	84815301	0	0	0	0	0	0	0	64196841
15	64196841	0	0	0	0	0	0	0	0
16-22	0	0	0	0	0	0	0	0	0

Table A.11: 23 step linear DP for SHA-256. There are 79 GH local collisions.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0	06410482	06410482	0	0	0	06410482	0	0	0
1	5a5956e6	4310a0e6	06410482	0	0	e282dbd7	06410482	0	0
2	936b23d5	22053aa0	4310a0e6	06410482	0	f7709310	e282dbd7	06410482	0
3	c4c16eb1	9421149d	22053aa0	4310a0e6	06410482	5d4bca94	f7709310	e282dbd7	06410482
4	50e078c8	b50c2248	9421149d	22053aa0	4310a0e6	f6fbb4b5	5d4bca94	f7709310	e282dbd7
5	d0d250ef	01606240	b50c2248	9421149d	22053aa0	4dff4081	f6fbb4b5	5d4bca94	f7709310
6	ce2982f4	4036003e	01606240	b50c2248	9421149d	f1e22908	4dff4081	f6fbb4b5	5d4bca94
7	6cb9d29e	8bc0502c	4036003e	01606240	b50c2248	561e3c0e	f1e22908	4dff4081	f6fbb4b5
8	e3a3f08f	0	8bc0502c	4036003e	01606240	17d8da6e	561e3c0e	f1e22908	4dff4081
9	540feff8	0	0	8bc0502c	4036003e	01606240	17d8da6e	561e3c0e	f1e22908
10	09d6a48d	0	0	0	8bc0502c	4036003e	01606240	17d8da6e	561e3c0e
11	b3d6fdee	0	0	0	0	8bc0502c	4036003e	01606240	17d8da6e
12	404eb561	0	0	0	0	0	8bc0502c	4036003e	01606240
13	01606240	0	0	0	0	0	0	8bc0502c	4036003e
14	4036003e	0	0	0	0	0	0	0	8bc0502c
15	8bc0502c	0	0	0	0	0	0	0	0
16-22	0	0	0	0	0	0	0	0	0

Table A.12: Colliding message pair for 18-step SHA-256 with standard IV. These two messages satisfy the conditions of Table 3.2. The initial perturbation w is taken to be `abcdef01`.

M_1	0-3	0	0	0	0
	4-7	ab3da9c1	f84ea2f0	0	0
	8-11	7133174a	feb9c846	0	0
	12-15	0	0	0	0
M_2	0-3	0	0	0	abcdef01
	4-7	00820670	a4bd669c	d65210fe	5a322501
	8-11	dcc10ab5	feb9c846	0	543210ff
	12-15	0	0	0	0

Table A.13: Colliding message pair for 18 step SHA-512 with standard IV. These two messages satisfy the conditions of Table 3.2. The initial perturbation w is taken to be `abcdef0123456789`.

M_1	0-1	0	0
	2-3	0	0
	4-5	1e4387c99964e3df	833c00f2b6a3adde
	6-7	0	0
	8-9	823dd46c0c9346e5	29c51252ea4e9d0b
	10-11	0	0
	12-13	0	0
	14-15	0	0
M_2	0-1	0	0
	2-3	0	abcdef0123456789
	4-5	3e1ae599c9a10350	d80a93fe0e9ecbef
	6-7	b0c053b0e95ee217	845204a29bf9d88f
	8-9	f5b543b1f125cc3e	29c51252ea4e9d0b
	10-11	0	543210fedcba9877
	12-13	0	0
	14-15	0	0

Table A.14: Colliding message pair for 20 step SHA-256 with standard IV.

M_1	0-3	0	0	0	f5db4e55
	4-7	ce1ed8cf	26586d76	a8a699ba	fa28bc9a
	8-11	acf3d949	72a3426f	e7f10e02	aaf3823e
	12-15	0	0	0	0
M_2	0-3	0	0	0	f5db4e55
	4-7	ce1ed8cf	26586d77	ad069839	f608bd1a
	8-11	33efcf9b	72a3426f	e7f10e02	aaf3823e
	12-15	0	ffffffff	0	0

Table A.15: Colliding message pair for 20-step SHA-512 with standard IV.

W_1	0-1	0	0
	2-3	0	b5d7924a5f1dfac8
	4-5	1d8a59931fde4139	f585791431ba6218
	6-7	4cb5bceeaf756795	3a68e5bec866399e
	8-9	b3ae5a6fb00c2fec	533c19c84c7c8969
	10-11	a28b08dc36fe38cf	aaf3823c2a004b1f
	12-13	0	0
	14-15	0	0
W_2	0-1	0	0
	2-3	0	b5d7924a5f1dfac8
	4-5	1d8a59931fde4139	f585791431ba6219
	6-7	4cb1fceeaef56794	3a74a5bec7e6399e
	8-9	14ab7440a4022222	533c19c84c7c8969
	10-11	a28b08dc36fe38cf	aaf3823c2a004b1f
	12-13	0	ffffffffffffffff
	14-15	0	0

Table A.16: Colliding message pair for 21-step SHA-512 with standard IV. The six free words are taken to be all zero in the first message.

W_1	0-1	0	0
	2-3	0	f31b03afd93b0f6d
	4-5	8308c99162dbe6fe	739cde771413fca7
	6-7	49404e11d70bcb89	b913ae32c4c3736f
	8-9	e284a20a666a735b	573c19c84c7c8968
	10-11	a28b08dc36fe38d0	6db010d6afe33679
	12-13	8d41a28b0d847692	0
	14-15	0	0
W_2	0-1	0	0
	2-3	0	f31b03afd93b0f6d
	4-5	8308c99162dbe6fe	739cde771413fca7
	6-7	49404e11d70bcb8a	b907ee32c543736f
	8-9	e290620a65ea735b	533c19c84c7c8969
	10-11	a28b08dc36fe38d0	6db010d6afe33679
	12-13	8d41a28b0d847692	0
	14-15	ffffffffffffffff	0

Table A.17: Colliding message pair for 21-step SHA-256 with standard IV. The six free words are taken to be all zero in the first message.

W_1	0-3	0	0	0	e9f8ba95
	4-7	534d9ceb	c6ceba8e	3b9d659d	e5eb4a91
	8-11	93bc685a	76a3426e	e7f10e03	b6d615fd
	12-15	8d41a28d	0	0	0
W_2	0-3	0	0	0	e9f8ba95
	4-7	534d9ceb	c6ceba8e	3b9d659e	ea0b4a10
	8-11	8f9c68d8	7663426f	e7f10e03	b6d615fd
	12-15	8d41a28d	0	ffffffff	0

Table A.18: Colliding message pair for 22 step SHA-256 with standard IV.

M_1	0-3	0	0	0be293bf	99c539c9
	4-7	1c672194	99b6a58a	5bf1d0ae	0a9a18d3
	8-11	0c18cf1c	329b3e6e	dc4e7a43	ab33823f
	12-15	8d41a28d	7f214e03	0	0
M_2	0-3	0	0	0be293bf	99c539c9
	4-7	1c672194	99b6a58a	5bf1d0ae	0a9a18d4
	8-11	07d56809	329b3e6e	dc0e7a44	ab33823f
	12-15	8d41a28d	7f214e03	0	ffffffff

Table A.19: Colliding message pair for 22 step SHA-512 with standard IV.

M_1	0-1	0	0
	2-3	c2bc8e9a85e2eb5a	6d623c5d5a2a1442
	4-5	cd38e6dee1458de7	acb73305cddb1207
	6-7	148f31a512bbade5	ecd66ba86d4ab7e9
	8-9	92aafb1e9cfa1fcb	533c19b80a7c8968
	10-11	e3ce7a41b11b4d75	aef3823c2a004b20
	12-13	8d41a28b0d847692	7f214e01c4e96950
	14-15	0	0
M_2	0-1	0	0
	2-3	c2bc8e9a85e2eb5a	6d623c5d5a2a1442
	4-5	cd38e6dee1458de7	acb73305cddb1207
	6-7	148f31a512bbade5	ecd66ba86d4ab7ea
	8-9	90668fd7ec6718ee	533c19b80a7c8968
	10-11	dfce7a41b11b4d76	aef3823c2a004b20
	12-13	8d41a28b0d847692	7f214e01c4e96950
	14-15	0	ffffffffffffffff

Table A.20: Colliding message pair for 23-step SHA-256 with standard IV. These messages utilize a single local collision starting at Step $i = 8$.

W_1	0-3	122060e3	000f813f	d92d3fc6	ea4a475f
	4-7	fb0c6581	dc4558c4	d86428b4	6e2ca576
	8-11	c8d597bf	6372d4c2	ddbd721c	79d654c4
	12-15	f0064002	a894b7b6	91b7628e	3224db20
W_2	0-3	122060e3	000f813f	d92d3fc6	ea4a475f
	4-7	fb0c6581	dc4558c4	d86428b4	6e2ca576
	8-11	c8d597c0	6372d4c1	ddbd721c	78d6b4c5
	12-15	f0064002	a894b7b6	91b7628e	3224db20

Table A.21: Colliding message pair for 23-step SHA-256 with standard IV. These messages utilize a single local collision starting at Step $i = 9$.

W_1	0-3	c201bef2	14cc32c9	3b80da44	d8212037
	4-7	8987161d	a790cb4a	53b8d726	89e9a288
	8-11	3edd76e0	05f41ddc	9ebc0fc3	e099698a
	12-15	2eaec58f	e7060b78	95d7030d	6bf777c0
W_2	0-3	c201bef2	14cc32c9	3b80da44	d8212037
	4-7	8987161d	a790cb4a	53b8d726	89e9a288
	8-11	3edd76e0	05f41ddd	9ebc0fc2	e099c98a
	12-15	2daf2590	e7060b78	95d7030d	6bf777c0

Table A.22: Colliding message pair for 24-step SHA-256 with standard IV. These messages utilize a single local collision starting at Step $i = 10$.

W_1	0-3	657adf63	06c066d7	90f0b709	95a3e1d1
	4-7	c3017f24	fad6c2bf	dff43685	6abff0da
	8-11	e6cfc63f	de8fb4c1	c20ca05b	f74815cc
	12-15	c2e789d9	208e7105	cc08b6cf	70171840
W_2	0-3	657adf63	06c066d7	90f0b709	95a3e1d1
	4-7	c3017f24	fad6c2bf	dff43685	6abff0da
	8-11	e6cfc63f	de8fb4c1	c20ca05c	f74815cb
	12-15	c2e7e9d9	1f8ed106	cc08b6cf	70171840

Table A.23: Colliding message pair for 23-round SHA-512 with standard IV. These messages utilize a single local collision starting at Step $i = 8$.

W_1	0-1	b9fa6fc4729ca55c	8718310e1b3590e1
	2-3	1d3d530cb075b721	99166b30ecbdd705
	4-5	27ed55b66c090b62	754b2163ff6feec5
	6-7	6685f40fd8ab08f8	590c1c0522f6fdfd
	8-9	b947bb4013b688c1	d9d72ca8ab1cac04
	10-11	69d0e120220d4edc	30a2e93aeef24e3f
	12-13	84e76299718478b9	f11ae711647763e5
	14-15	d621d2687946e862	0ee57069123ecc8b
W_2	0-1	b9fa6fc4729ca55c	8718310e1b3590e1
	2-3	1d3d530cb075b721	99166b30ecbdd705
	4-5	27ed55b66c090b62	754b2163ff6feec5
	6-7	6685f40fd8ab08f8	590c1c0522f6fdfd
	8-9	b947bb4013b688c2	d9d72ca8ab1cac03
	10-11	69d0e120220d4edc	30a3493aeef25076
	12-13	84e76299718478b9	f11ae711647763e5
	14-15	d621d2687946e862	0ee57069123ecc8b

Table A.24: Colliding message pair for 24-round SHA-512 with standard IV. These messages utilize a single local collision starting at Step $i = 10$.

W_1	0-1	dedb689cfc766965	c7b8e064ff720f7c
	2-3	c136883560348c9c	3747df7d0cf47678
	4-5	855e17555cfedc5f	88566babccaa63e9
	6-7	5dda9777938b73cd	b17b00574a4e4216
	8-9	86f3ff48fd12ea19	cd15c6f8d6da38ce
	10-11	5e2c6b7b0411e70b	36ed67e93a794e66
	12-13	1b65e96b02767821	04d0950089db6c68
	14-15	5bc9b9673e38eff3	b05d879ad024d3fa
W_2	0-1	dedb689cfc766965	c7b8e064ff720f7c
	2-3	c136883560348c9c	3747df7d0cf47678
	4-5	855e17555cfedc5f	88566babccaa63e9
	6-7	5dda9777938b73cd	b17b00574a4e4216
	8-9	86f3ff48fd12ea19	cd15c6f8d6da38ce
	10-11	5e2c6b7b0411e70c	36ed67e93a794e65
	12-13	1b66096b02767829	04d0f50089db6e9f
	14-15	5bc9b9673e38eff3	b05d879ad024d3fa

Table A.25: Test vectors for SShash. The given digests are the outputs of SShash on the string "aaa".

SShash-256	0x27ef472acd480e556be88c4b320008b2 78d1819fe297abdd97ed947a295e3eb4
SShash-512	0x47cd2dafcc070c317d242d027a2b3ec6 5345065dadbeff05cf88745a2a759c8f ff67b38965dbd9fbc15280fe41415b83 64fe8f46baf9dc60a5173912480ce916

A.4 Compression Function for SShash-256

The required macros and the description of the compression function for SShash-256 are provided below. Corresponding macros for SShash-512 are similar and hence not provided. Test vectors for SShash-256 and SShash-512 are given in Table A.25.

```
#define word32 unsigned int

#define ROTR(x, n) (((x) >> (n)) | ((x) << (32-(n))))
#define SHR(x, n)   ( (x) >> (n) )

#define NEW_SIG_0(x) ( ((ROTR(x,2))^(ROTR(x,13))^(ROTR(x,22)) )\
                        ^ 0xdcb2344c )
#define NEW_SIG_1(x) ( ((ROTR(x,6))^(ROTR(x,11))^(ROTR(x,25)) )\
                        ^ 0x9b097671 )

#define sig_0(x) ( (ROTR(x,7)) ^ (ROTR(x,18)) ^ (SHR(x,3)) )
#define sig_1(x) ( (ROTR(x,17)) ^ (ROTR(x,19)) ^ (SHR(x,10)) )

#define CH(x,y,z) ((z)^((x)&((y)^(z))))
#define MAJ(x,y,z) (((x)&(y))|((z)&((x)|(y))))

#define a1(i) T[(0-i)&7]
#define b1(i) T[(1-i)&7]
#define c1(i) T[(2-i)&7]
#define d1(i) T[(3-i)&7]
#define e1(i) T[(4-i)&7]
#define f1(i) T[(5-i)&7]
#define g1(i) T[(6-i)&7]
#define h1(i) T[(7-i)&7]
```

```
#define FF_XOR_START(T) { \
 T[0]^=a1(i); T[1]^=b1(i); T[2]^=c1(i); T[3]^=d1(i); \
 T[4]^=e1(i); T[5]^=f1(i); T[6]^=g1(i); T[7]^=h1(i); \
}

#define FF_ADD_START(T) { \
 T[0]+=a1(i); T[1]+=b1(i); T[2]+=c1(i); T[3]+=d1(i); \
 T[4]+=e1(i); T[5]+=f1(i); T[6]+=g1(i); T[7]+=h1(i); \
}

#define FF_ADD(T) {\
 a1(i) +=T[0]; b1(i) +=T[1]; c1(i) +=T[2]; d1(i) +=T[3]; \
 e1(i) +=T[4]; f1(i) +=T[5]; g1(i) +=T[6]; h1(i) +=T[7]; \
 T[0]=a1(i); T[1]=b1(i); T[2]=c1(i); T[3]=d1(i); \
 T[4]=e1(i); T[5]=f1(i); T[6]=g1(i); T[7]=h1(i); \
}

#define FF_XOR(T) { \
 a1(i) ^=T[0]; b1(i) ^=T[1]; c1(i) ^=T[2]; d1(i) ^=T[3]; \
 e1(i) ^=T[4]; f1(i) ^=T[5]; g1(i) ^=T[6]; h1(i) ^=T[7]; \
 T[0]=a1(i); T[1]=b1(i); T[2]=c1(i); T[3]=d1(i); \
 T[4]=e1(i); T[5]=f1(i); T[6]=g1(i); T[7]=h1(i); \
}

#define FF_FINAL(REG) { \
 REG[7] += h1(0); REG[6] += g1(0); \
 REG[5] += f1(0); REG[4] += e1(0); \
 REG[3] += d1(0); REG[2] += c1(0); \
 REG[1] += b1(0); REG[0] += a1(0); \
}

#define R_0(i) { \
 temp1 = NEW_SIG_1(e1(i))+CH(e1(i),f1(i),g1(i))+K[i]; \
 temp2 = NEW_SIG_0(a1(i)) + MAJ(a1(i),b1(i),c1(i)); \
 d1(i) += temp1 + h1(i); d1(i) ^= data[i]; \
 h1(i) ^= temp1 + temp2 + data[i]; \
}

#define R_1(i) { \
 temp1 = NEW_SIG_1(e1(i))+CH(e1(i),f1(i),g1(i))+K[i+16]; \
 temp2 = NEW_SIG_0(a1(i)) + MAJ(a1(i),b1(i),c1(i)); \
 d1(i) += temp1 + h1(i); d1(i) ^= W[i+16]; \
 h1(i) ^= temp1 + temp2 + W[i+16]; \
}

#define R_2(i) { \
```

```
 temp1 = NEW_SIG_1(e1(i))+CH(e1(i),f1(i),g1(i))+K[i+32]; \
 temp2 = NEW_SIG_0(a1(i)) + MAJ(a1(i),b1(i),c1(i)); \
 d1(i) += temp1 + h1(i); d1(i) ^= W[i+32]; \
 h1(i) ^= temp1 + temp2 + W[i+32]; \
}

#define R_3(i) { \
 temp1 = NEW_SIG_1(e1(i))+CH(e1(i),f1(i),g1(i))+K[i+48]; \
 temp2 = NEW_SIG_0(a1(i)) + MAJ(a1(i),b1(i),c1(i)); \
 d1(i) += temp1 + h1(i); d1(i) ^= W[i+48]; \
 h1(i) ^= temp1 + temp2 + W[i+48]; \
}

compression_func(word32 *reg, const word32 *data){
  word32 i, W[64], temp1, temp2, T[8];
  static word32 T1[8]={0,0,0,0,0,0,0,0}, T2[8]={0,0,0,0,0,0,0,0};

  memcpy(W,data,64);
  for(i=16; i<64; i++)
    W[i] = sig_1(W[i-2])+W[i-7]+sig_0(W[i-15])+W[i-16];

  memcpy(T, reg, sizeof(T));

  {
    R_0( 0); R_0( 1); R_0( 2); R_0( 3);
    R_0( 4); R_0( 5); R_0( 6); R_0( 7); FF_XOR_START(T1);
    R_0( 8); R_0( 9); R_0(10); R_0(11);
    R_0(12); R_0(13); R_0(14); FF_ADD_START(T2); R_0(15);
  }
  {
    R_1( 0); R_1( 1); R_1( 2); R_1( 3);
    R_1( 4); R_1( 5); R_1( 6); R_1( 7); FF_ADD(T1);
    R_1( 8); R_1( 9); R_1(10); R_1(11);
    R_1(12); R_1(13); R_1(14); FF_XOR(T2); R_1(15);
  }
  {
    R_2( 0); R_2( 1); R_2( 2); R_2( 3);
    R_2( 4); R_2( 5); R_2( 6); R_2( 7); FF_XOR(T1);
    R_2( 8); R_2( 9); R_2(10); R_2(11);
    R_2(12); R_2(13); R_2(14); FF_ADD(T2); R_2(15);
  }
  {
    R_3( 0); R_3( 1); R_3( 2); R_3( 3);
    R_3( 4); R_3( 5); R_3( 6); R_3( 7); FF_ADD(T1);
    R_3( 8); R_3( 9); R_3(10); R_3(11);
    R_3(12); R_3(13); R_3(14); FF_XOR(T2); R_3(15);
```

```
  }
  FF_FINAL(reg);
}
```